EXPERT ADVISOR

DOS

Up to and including DOS 4.0

Jonathan Kamin

Overviews · Procedures · Examples · Warnings · Tips

Addison-Wesley Publishing Company

FREE QUICK REFERENCE GUIDE INSIDE!
A $7.95 retail value

EXPERT ADVISOR

DOS

Lindsey Grant
5505 Durbin Road
Bethesda, MD 20814
(301) 652-8940

1/91

EXPERT ADVISOR

DOS

Jonathan Kamin

Addison-Wesley Publishing Company, Inc.
Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial capital letters. (i.e., Microsoft is a registered trademark of Microsoft Corporation).

Apple, Mac, and Macintosh are registered trademarks of Apple Computer, Inc. AutoCAD is a trademark of Autodesk Systems. dBASE, dBASE III PLUS, and Framework are trademarks of Ashton-Tate, Inc. IBM is a registered trademark of International Business Machines Corporation. Proprinter, Quietwriter, Data Migration Facility, PS, BASICA, and Personal Computer AT are trademarks of International Business Machines Corporation. MS is a registered trademark of Microsoft Corporation. MS-DOS and QuickBASIC are trademarks of Microsoft Corporation. SideKick, SuperKey, and Turbo BASIC are trademarks of Borland International. Lotus, Symphony, and 1-2-3 are registered trademarks of Lotus Development Corporation. WordPerfect is a trademark of WordPerfect Corporation. DESQ and DESQview are trademarks of Quarterdeck Office Systems. R:Base is a trademark of MicroRim, Inc. Ventura is a trademark of Xerox Corporation. SuperCalc is a trademark of Computer Associates International. WordStar is a registered trademark of MicroPro International. Epson and GrafTrax are trademarks of Epson America, Inc. GEM is a trademark of Digital Research, Inc. Smartkey is a trademark of FBN Software.

Library of Congress Cataloging-in-Publication Data
Jonathan Kamin
Expert advisor: DOS.
Includes indexes.
1. DOS (Computer program) I. Title.
T385.C542 1988 604.2'4'02855369 88-3346
ISBN 0-201-51743-4 (pbk.)

Copyright © 1989 by Jonathan Kamin

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. Published simultaneously in Canada.

Sponsoring Editor: Carole Alden
Series Editor: Tanya Kucak
Technical Reviewer: Michael J. Eager
Cover design by Corey & Company: Designers
Text design by Joyce Weston
Set in 10.5-point Palatino by Publication Services, Inc.

ABCDEFGHIJ-AL-89
First printing, May, 1989

To the memory of
Judith Liff

Acknowledgments

Before anything else, I wish to thank my wife, Nancy, and my sons, Jaron and Jordan, for their patience and forbearance while this Herculean labor was carried out. For them, it required pretending I wasn't there for many long periods, and they all took it well. If you like this book, you should thank them too.

More immediately related to the contents, many people contributed directly and indirectly to the completion of this book. Debra Globe of Compaq Computer Corporation saw to it that I had the latest release of DOS to work with. Therese Myers and Dan Sweeney of Quarterdeck Office Systems gave me access to and help with the DESQview environment, which allowed me to complete many aspects of the project simultaneously. Ola Bennett kindly lent me her Apple Macintosh® computer to create some of the illustrations. Gale Steiner and Coleen Haworth of Digital Research provided me with GEM Draw Plus and some ancillary materials, with which I created the remainder of the illustrations. Linda Devan, Bill Amanna, and Bob Latchford of IBM dug out some obscure technical details on the latest releases of PC-DOS and IBM DOS. Joe Campbell and Martin Waterhouse gave me explanations of technical points that would otherwise have been beyond my understanding. Zelda Gifford was constantly available as a sounding board during the creation of the manuscript and contributed greatly to its coherence and clarity.

Of course the project would not have been possible without the support of many people at Addison-Wesley. Carole Alden talked me into doing the book in the first place, and ran interference when necessary. Alexandra McDowell and Rachel Guichard provided constant editorial and moral support. Thanks are also due to Michael Eager for reviewing the manuscript for technical accuracy; Maeve A. Cullinane and Perry McIntosh for coordinating the production; Susan Lane Riley and Jane Kilgore for editing the copy; Richard Cook at Publication Services for typesetting; and C. J. Shane for preparing the index.

Contents

Introduction xi

DOS Versions xiv

Alphabetical Entries 1

Appendix A: ANSI Escape Sequences 506

Appendix B: ASCII Tables 509

Appendix C: List of Tables 517

Appendix D: List of Figures 518

Appendix E: List of Programs 521

General Index 523

Introduction

Expert Advisor: DOS is a complete reference to all DOS releases from 2.0 through 4.0 for any user. The beginning user will quickly learn how to use DOS commands without wading through a lengthy tutorial or volumes of descriptive text. The intermediate user will find quick reminders concerning specific commands and processes. The advanced user will find documentation of obscure commands, new ways to use familiar commands, and unusual ways to combine commands to complete complex tasks easily. Any DOS user will find this book a valuable learning tool and a rich source of pointers, reminders, and instructions.

The book contains every command in PC-DOS from release 1.0 through 3.3, IBM DOS 4.0, and every command in the standard version of MS-DOS from release 2.1 through 4.0. (Since each computer manufacturer's version of MS-DOS has unique features, you may find that your version includes a few commands or command options not covered here. Conversely, your version may not include some of the standard commands.) Many entries include batch programs that allow you to unlock the full power of DOS commands. In addition to DOS commands, you'll find entries on such related topics as the organization of computer memory, the structure of your disks, special backup techniques, and graphics.

All the DOS commands and general topics are arranged alphabetically and cross-referenced so that you will be able to find related information quickly and easily. You'll also find a generous amount of cross-referencing in the body of the text. Because of this careful organization, it's easy to flip to appropriate topics to find additional information.

Here is what you will find in each alphabetical entry:

Overview The Overview provides a description in simple terms of what a command does. You will find this section helpful if you have forgotten a command's function, or want to learn new and unfamiliar commands.

Procedure This section provides detailed step-by-step instructions for using each command in all its forms. Many DOS commands have a wealth of switches and parameters. These parameters, used together carefully, can

let you use commands in ways quite different from their obvious purpose. If you remember a command but have forgotten how to use it, this section provides a review.

Examples In the Examples section, you will find many practical uses for each command, some of which you might not have thought of. Many of the commands are illustrated with batch files that incorporate the specific command with unique combinations of switches, and with other commands to create useful, time-saving tools.

Procedure/ Examples Some commands cannot be understood easily without an immediate illustration, but perform a relatively limited function. For such commands, the procedures and simple examples have been combined.

Warnings DOS is not free of pitfalls and problems. In fact, it's sometimes downright hostile. In the Warnings section, you will find out where you might encounter problems, unintended and undesirable results, and major disasters. When you're lucky, you will also be presented with solutions. At worst, you will be told how to avoid these pitfalls.

Tips The Tips section will be of interest to readers who want to expand their knowledge of DOS. Here you will find related commands that may accomplish a task more effectively, as well as tricks, hints, and insights that will enhance your productivity as a computer user.

In the back of the book you will find five appendices. Appendix A presents the escape sequences used by the ANSI.SYS device driver to control the keyboard and screen. Appendix B contains a complete ASCII table and a table of extended ASCII codes for the keys that do not produce ASCII characters. Appendix C contains a list of tables, Appendix D is a list of figures, and Appendix E is a list of programs. The appendices are followed by the General Index, which lists each command, topic, and message discussed in the book. Even though you will be able to find most of these subjects listed alphabetically, some topics are embedded within the text, and the General Index will help you find these topics.

Syntax Many commands offer a plethora of switches and options. In each entry for which there is more than one form of command syntax, all the syntax forms appear at the beginning of the entry, with identifying numbers. For example, when a syntax form is discussed in the body of the text, the form is again presented with its identifying number before the discussion. In the syntax forms, anything optional is enclosed in brackets. Generic references to drives, paths, file names, and other parameters are in *italics*. File names appear in uppercase, except when they appear as parts of commands or syntax forms. The file name *filename.ext* represents any legal DOS file name.

This book is much more than an encyclopedic reference to DOS. Beyond descriptions and examples of commands, you will find a wealth of information that can only be gained from years of using and writing about DOS on an advanced level. I hope you will find *Expert Advisor: DOS* an information source that serves you as long as you work with DOS.

DOS Versions

Where applicable, all versions of MS-DOS and PC-DOS are listed at the beginning of each entry. There have been many releases of DOS, each slightly different from the others. New commands have been introduced with each release, and each manufacturer offers a slightly different selection of commands and switches. In light of this variety, the following conventions have been used:

1. When something is true of all versions of DOS, the discussion will refer to DOS.
2. When something is true of all subreleases of a major release, both in MS-DOS and PC-DOS, the discussion will refer to DOS 2, DOS 3, or DOS 4.
3. When something is true of only one release, or only a few, the releases will be mentioned specifically; for example, MS-DOS 3.1 to 3.31, or PC-DOS 3.0.

Alphabetical Entries

Abort, Retry, Ignore, Fail?

Overview

The *Abort, Retry, Ignore?* prompt appears when DOS experiences a *device error* while attempting to get input from or write to a device other than the console (the keyboard and screen). This prompt indicates that DOS has attempted unsuccessfully to complete the requested task four times and is asking you what to do next.

If you press A for *Abort*, the current program ends, and you are returned to the DOS prompt.

If you press R for *Retry*, DOS tries to complete the task four more times. If it is unsuccessful, the same error message and prompt reappear.

If you press I for *Ignore*, DOS continues as if the error had not occurred. The application program you are using will not know that an error has occurred and will assume the operation was successfully completed. If DOS considers this option to be too dangerous, it will abort the operation and the program and return you to the DOS prompt.

DOS 3.3 has added a fourth option—*Fail*—to the prompt. *Fail* either supplements or replaces the *Ignore* option, depending on the error. If you press F for *Fail*, DOS proceeds as if the error had not occurred, but sends a message indicating the failure to the application program.

Procedure

The best response depends on the type of error and the situation in which the error occurs. Generally, disk errors are most critical. Table 1 lists both the more common messages that include this prompt and some general guidelines for responding to it.

Warnings

DOS keeps in memory a copy of the file allocation table of any disk it is currently using. Under normal circumstances, before reading or writing to a disk DOS checks that its file allocation table matches the one in memory. Thus, it knows whether you have changed disks and will read and write correctly. However, the Abort, Retry, Ignore/Fail? prompt indicates that circumstances are not normal and that DOS does *not* know if you have changed disks. If you change disks and retry, DOS may write the file allocation table in memory over the file allocation table on the disk. If the disk contains data, some of it will be overwritten and the rest will be lost. The only way to recover the data is to run the Recover program (see Recover) on the entire disk.

A ABORT, RETRY, IGNORE, FAIL?

When the Abort, Retry, Ignore/Fail? prompt accompanies a message not discussed here, always choose Abort. In such cases, the problem is caused by the program and cannot be easily fixed by the user. Inform the publisher of the program that caused the error message and ask for help.

TABLE 1 **Device Errors**

ERROR MESSAGE	PROBABLE CAUSE	RECOMMENDED ACTION
Printer Problems		
`Error writing device PRN`	Printer off-line (not turned on) or out of paper	Correct the condition and retry
`No Paper`	Printer out of paper	Put paper in the printer and retry
Disk/Diskette Problems		
`General failure reading/writing drive d`	Diskette is not formatted	Replace with a formatted diskette and retry; abort if this is not possible.
	On some computers, drive speed is not synchronized with CPU's turbo speed.	Abort, then slow down your CPU speed and discuss a BIOS transplant with your dealer. Some clone BIOS versions are prone to this problem.
`Non-DOS disk`	File allocation table is damaged or contains invalid information.	The diskette is damaged. If reading, first retry with a different disk if possible; otherwise, abort. Next, run ChkDsk on the damaged disk; if necessary, run Recover as well.
`Read fault error reading drive d`	DOS was unable to read from the disk; disk may be improperly seated in the drive or may have a bad sector.	Open the drive, take out the diskette and reinsert it, and retry. If the failure repeats, abort and run ChkDsk on the disk.

Table 1 continues

ERROR MESSAGE	PROBABLE CAUSE	RECOMMENDED ACTION
`Write fault error writing drive d`	DOS was unable to write to the disk; disk may be improperly seated in the drive, or may have a bad sector.	Open the drive, take out the diskette and reinsert it, and retry. If the failure repeats, abort and run ChkDsk on the disk.
`Sector not found error`	The data DOS is trying to read is probably in a bad sector.	The diskette is damaged. If reading, first retry with a different disk if possible; otherwise, abort. Next, run ChkDsk on the damaged disk; if necessary, run Recover as well.
`Drive not ready error`	You forgot to close the door on the diskette drive.	Close the door and retry.
	A diskette drive is current, but you removed the diskette from it and tried to make another drive current.	Press I for *Ignore* until you see the message *Current drive is no longer valid*, then make your preferred drive current.
`Seek error reading/writing drive d`	DOS was unable to locate the proper track on a disk.	If it's a diskette, open the drive, take out the diskette and reinsert it, and retry. If the failure repeats, abort and run ChkDsk on the disk. If it's a hard disk, abort and run ChkDsk. If this message occurs often, back up your hard disk completely and get a new one.
`Data error reading/writing drive d`	DOS is probably attempting to read or write to a bad sector.	The diskette is damaged. If reading, first retry with a different disk if possible; otherwise, abort. Next, run ChkDsk on the damaged disk; if necessary, run Recover as well.

Table 1 continues

ERROR MESSAGE	PROBABLE CAUSE	RECOMMENDED ACTION
`Write protect error writing drive d`	There is a write-protect tab on your diskette.	Take out the diskette and read the label. You probably write-protected the diskette for a good reason. If it's the right diskette, remove the write-protect tab and retry.
Other Problems		
`Sharing violation reading drive d`	Someone else on your network is using the file you want to use in a mode that does not allow you to use it at the same time.	Wait awhile, then retry. If you can't wait, abort.
`Lock violation reading/writing drive d`	A program tried to access a part of a file in use by another program.	Wait awhile, then retry. If you can't wait, abort.
`FCB unavailable`	All the file control blocks (see FCBs) available in your system are in use.	Abort. Rewrite your CONFIG.SYS file (see CONFIG.SYS File) to include a FCBs = command with a higher parameter value. Reboot the computer.

Allocation Units. *See* Disk Clusters

ANSI.SYS

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

ANSI.SYS is a device driver designed to make your screen and keyboard compatible with the standards established for terminals by the American National Standards Institute (ANSI). With some non–IBM-compatible

computers, ANSI.SYS must be loaded for the computer to function correctly. Also, some software does not function properly unless ANSI.SYS is loaded. Software packages that require the ANSI.SYS driver usually say so and explain how to load it in the documentation.

For most PCs and compatibles, however, the principal purpose of the ANSI.SYS driver is to give you access to *extended screen and keyboard functions*. The screen functions allow you to place the cursor on the screen, erase text on the screen, set the video mode, and establish video attributes. The video attributes include highlighting, underlining, and inverse video on monochrome screens, as well as foreground (text) and background color combinations on color screens. (You cannot set a border color with these functions.) The extended keyboard functions let you reassign the position of keys on the keyboard or assign strings—groups of characters—to particular keys.

These functions are included primarily for programmers to control the screen and keyboard in BASIC and other programming languages. However, nonprogrammers can use these functions for the same purposes in batch programs and text files.

Procedure

Loading the Driver To have access to the extended screen and keyboard functions, load the ANSI.SYS driver by using the command `device=ansi.sys` in your CONFIG.SYS file. For this command to work, the ANSI.SYS file (found on the DOS system diskette) must be present in the computer, either on the boot disk or on another disk that DOS can access. If the ANSI.SYS file is not on the boot disk, you must specify the drive and full path name of the directory on which it appears.

Thus, to boot from a diskette when ANSI.SYS is on a hard disk designated C in a directory called \DOS\BOOT, enter the command `device=c:\dos\boot\ansi.sys` in your CONFIG.SYS file. (Of course, if you can't read your hard disk until a device driver for it is loaded, this line must follow the line that loads the hard disk driver.)

The DOS 4.0 version of ANSI.SYS can be loaded with one of three optional switches. The command `device=ansi.sys /x` will cause ANSI.SYS to regard the separate cursor and numeric-keypad keys on the extended (101-key) keyboard as separate sets of keys. Thus, you can make separate sets of key assignments to each of them. On the other hand, `device=ansi.sys /k` will tell ANSI.SYS to treat the equivalent cursor and numeric keys as though they were the same key, so that, for example, a reassignment of the Home

key on the cursor keypad would also affect the 7 key on the numeric keypad. You can override these switches with escape sequences once the driver is loaded.

A third switch, /l, interacts with the Mode command to maintain the number of rows of text established by that command even when an application tries to force the screen to its default of 25 rows. This switch has an effect only if you have an EGA or VGA video card and an appropriate monitor, and often has no effect anyway. (See Mode.)

Using the Driver After loading the driver, you can use the extended screen and keyboard functions. Activate these functions by calling them with escape sequences. An escape sequence is a specific series of characters that functions as a command to a peripheral device. (You are probably most familiar with escape sequences for controlling your printer.) Most escape sequences begin with an escape character (ASCII 27 or Ctrl-[ ). Most ANSI escape sequences have the following four parts:

1. An escape character (ASCII 27 or Ctrl-[ )
2. A left-bracket character ([ )
3. One or more numeric parameters separated by semicolons
4. A one-letter command code that defines the effect of the escape sequence

Some escape sequences do not have numeric parameters, and others may require additional characters. Figure 1 illustrates an ANSI escape sequence.

There are three ways to pass escape sequences to the ANSI.SYS driver:

- Place them in an ASCII text file, and then use the Type command to display the file.
- Place them in a batch file in an Echo statement.
- Place them in a batch file in a Prompt statement.

See Type, Echo, and Prompt for more information about how these commands affect the screen display.

The first method is probably the easiest, though the second method is useful for creating fancy batch files. (For example, you can use escape

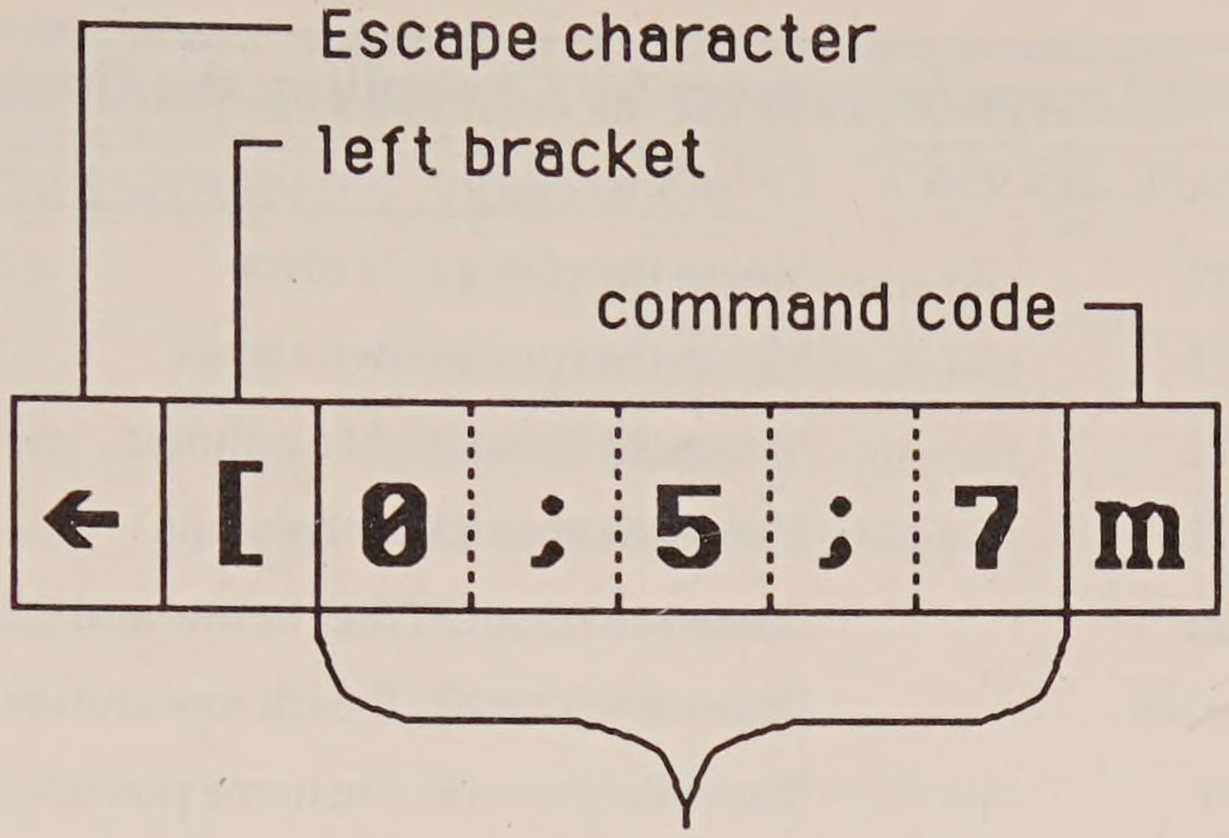


Figure 1: An ANSI escape sequence

sequences to place a message in flashing text at the center of the screen.) You can also use Prompt statements in the same manner (see Prompt). However, if you use a Prompt statement this way, you may have to issue another Prompt statement at the end of the batch file to restore the original prompt.

You cannot enter escape sequences on the command line because DOS cancels the command on the command line when you press either the Escape key or the Ctrl-[combination. You can use escape sequences in a Prompt statement that you type on the command line if you use the meta string $e in place of the escape character (see Prompt). But it's usually easier to place the escape sequences in a file you create with an editor or word processor.

There are four different groups of escape sequences. One set positions the cursor and erases text on the screen; a second set controls the video mode; a third sets screen colors and other video attributes, and the fourth set redefines the meaning of keys on the keyboard.

Controlling the Cursor The escape sequences shown in Table 2 place the cursor in particular rows and columns on the screen, move the cursor relative to its present position, clear individual lines or the entire screen, and memorize and restore the cursor position. In Table 2, *r* represents the horizontal row on the screen, and *c* represents the vertical column.

TABLE 2 **ANSI Escape Sequences for Controlling the Cursor**

ESCAPE SEQUENCE	EFFECT
←[*r*A	Moves the cursor up *r* rows
←[*r*B	Moves the cursor down *r* rows
←[*c*C	Moves the cursor right *c* columns
←[*c*D	Moves the cursor left *c* columns
←[*r*;*c*f	Places the cursor in the *r*th row and *c*th column
←[*r*;*c*H	Places the cursor in the *r*th row and *c*th column
←[H	Places the cursor in the home position (upper left corner of the screen)
←[2J	Clears the screen, leaving the cursor in the home position
←[K	Erases the current line from the current position of the cursor to the end
←[s	Saves (records) the current position of the cursor on the screen
←[u	Returns the cursor to the position it was in when the ←[s sequence was issued.

Setting the Video Mode The escape sequences in Table 3 are equivalent to specific parameters of the Mode command (see Mode). Escape sequences using the *h* terminating code are called *set-mode* sequences, and those ending in the *l* code are called *reset-mode* sequences. Their effect is the same, although you should use the *h* code the first time you issue a video mode sequence.

The sequence ←[?7l affects the display of lines at the DOS prompt (that is, when you issue a Type command or type on the command line). Lines exceeding 80 characters are displayed so that each character after 79 appears in column 80. The default condition is that lines exceeding 80 characters automatically wrap to the following screen line. To restore the default condition, issue the sequence ←[?7h. There is no equivalent Mode command for these escape sequences.

TABLE 3

ANSI Escape Sequences to Set the Video Mode

ESCAPE SEQUENCE	VIDEO MODE
←[=h	40 × 25 black and white text
←[=l	40 × 25 black and white text
←[=0h	40 × 25 black and white text
←[=0l	40 × 25 black and white text
←[=1h	40 × 25 color text
←[=1l	40 × 25 color text
←[=2h	80 × 25 black and white text
←[=2l	80 × 25 black and white text
←[=3h	80 × 25 color text
←[=3l	80 × 25 color text
←[=4h	320 × 200 4-color graphics
←[=4l	320 × 200 4-color graphics
←[=5h	320 × 200 black and white graphics
←[=5l	320 × 200 black and white graphics
←[=6h	640 × 200 black and white graphics
←[=6l	640 × 200 black and white graphics
←[=14h	640 × 200 16-color graphics*
←[=14l	640 × 200 16-color graphics*
←[=15h	640 × 350 monochrome graphics*
←[=15l	640 × 350 monochrome graphics*
←[=16h	640 × 350 16-color graphics*
←[=16l	640 × 350 16-color graphics*
←[=17h	640 × 480 2-color graphics*
←[=17l	640 × 480 2-color graphics*
←[=18h	640 × 480 16-color graphics*
←[=18l	640 × 480 16-color graphics*
←[=19h	320 × 200 256-color graphics*
←[=19l	320 × 200 256-color graphics*

** Available with DOS 4.0 and EGA or VGA only.*

Setting Video Attributes *Video attributes* affect the appearance of text on the screen. On a monochrome screen, text may have the following attributes:

Normal
High-intensity (boldface)
Underlined
Blinking
Inverse
Concealed

ANSI escape sequences can switch among all these display styles. On a color screen (color graphics, enhanced graphics, MCGA, or VGA), ANSI escape sequences can control foreground and background color and all the monochrome attributes except underlining.

The escape sequence for video attributes is

←[*n*; . . . ;*n*m

Values for *n* are taken from Table 4. Note that any number of parameters separated by semicolons can be included in a single escape sequence. Do not place a semicolon between the last parameter and the m control code.

Redefining Keys The escape sequence for redefining keys takes the form

←[*n*;*n*; . . . . *n*p

TABLE 4 **Parameters for ANSI Video Attribute Escape Sequences**

FOREGROUND	COLOR	BACKGROUND	OTHER ATTRIBUTES
30	Black	40	0 All attributes off
31	Red	41	1 High-intensity text
32	Green	42	
33	Yellow	43	
34	Blue	44	4 Underlined text
35	Magenta	45	5 Blinking text
36	Cyan	46	
37	White	47	7 Inverse video
			8 Concealed text

This sequence requires at least two parameters. First, enter the ASCII code (see ASCII Code) or extended ASCII code (see ASCII Code, Extended) of the key you want to define, followed by a semicolon. After the semicolon, enter the parameters that constitute the new definition for the key. The parameters may be decimal representations of ASCII values, strings enclosed in quotation marks, or any combination of both.

To clear a key assignment, reassign the key to itself by repeating the code of the assigned key; for example,

```
←[66;66p ←[0;68;0;68p
```

These two sequences assign the uppercase B to uppercase B, and the F10 key to the F10 key, respectively.

Miscellaneous Sequences An additional escape sequence available in DOS 4.0 allows you to access the extra keys on the extended (101-key) keyboard even if you didn't load the driver with the /x switch. The sequence ←[0q has the same effect as loading the driver with the /x switch, and the sequence ←[1q has the same effect as loading the driver with the /k switch. Thus, you can switch between standard and extended keyboard reassignments no matter how you loaded the driver.

Device Status Report is a special sequence that determines the current position of the cursor. When you issue the sequence ←[6n, the Cursor Position Report sequence appears on the screen. This sequence, which takes the form ←[*r*;*c*R, shows the row and column at which the cursor was located when you issued the Device Status Report sequence.

Examples

The ANSI escape sequences that affect the screen are useful for composing menus and help screens and for displaying messages in batch files.

Enhancing Text Displays The following text could be included in a batch file to display a message in inverse video, centered on the screen, with the first word flashing:

```
Echo ←[20C←[5;7m ERROR: ←[0;7mDrive letter must be specified ←[0m
```

The sequence ←[20C moves the cursor forward 20 spaces from the left edge of the screen. The escape sequence ←[5;7m at the beginning of the text turns on both inverse video and blinking. Since the text is 39 characters long, 21 spaces remain to the right of the message.

The next sequence, ←[0;7m, has two effects. The 0 parameter turns off all video attributes, which turns off both blinking and inverse video. The

7 parameter turns inverse video back on. This parameter is necessary because you can't turn off blinking without turning off all attributes.

The final sequence, ←[0m, at the end of the message turns off all the special video attributes again so that any following text will be in normal mode. Note that on a color screen this sequence results in light gray text on a black background. You can change the color combination by including the appropriate colors in an Echo or Prompt statement in a batch file.

Swapping Key Positions Suppose you recently switched from a standard PC to a PC AT. The Escape key on a PC is at the upper left corner of the main keyboard above the Tab key. On a PC AT, however, the Escape key is at the upper left corner of the numeric keypad. You can issue the following escape sequence if you want the Escape key to be in its familiar position:

```
←[27;96p ←[96;27p
```

This sequence enables the accent grave (`)/ tilde (~) key to function as the Escape key, and the Escape key to print the accent grave. However, pressing the shifted accent grave key would still print the tilde (~) because you have not reassigned the ASCII code for the tilde. Note that these sequences have no effect, and the keys behave normally, when you use WordStar or SideKick with these keys reassigned.

The following sequences swap the colon and the semicolon, both on the same key:

```
←[58;59p ←59;58p
```

This change can be helpful at the operating-system level because you constantly use the colon to refer to drives. Few DOS commands, however, use the semicolon.

To restore the original key definitions, issue the sequence

```
←[27;27p ←[96;96p ←[58;58p ←59;59p
```

The escape sequences that control the keyboard have several purposes. The following sequences assign several of the special symbols (all of which are ASCII characters with codes above 127) to the function keys:

```
←[0;59;201p ←[0;60;187p ←[0;61;200p
←[0;62;188p ←[0;63;186p ←[0;64;205p
```

Menus and Help Screens You can combine both screen and keyboard sequences to create more complex menus. Let's suppose you use a variety of combinations of resident utilities for various tasks with different software packages. (See TSRs.) If you don't always start the day with the

same program and don't want to memorize the order in which the TSRs must be loaded, you need a menu to choose among your possible configurations. The following batch program displays such a menu and assigns the names of batch files that load the TSR groups to various function keys:

TSR.BAT: A TSR Menu Program

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file.
`type tsr.mnu`	Display menu, reassign keys.
`prompt Choose by Function Key:`	Display a prompt to go with the screen.

The menu, which you display by using the Type command, appears in Figure 2. The code to generate the menu is the TSR.MNU file. Note the extensive use of ANSI escape sequences to format the menu. For

```
                    CONFIGURATION MENU

    [F1]  Help with this menu

    [F2]  DOS GRAPHICS Utility

    [F3]  Software Carousel

    [F4]  Microsoft Windows

    [F5]  Keyboard Enhancer only

    [F6]  Keyboard Enhancer, Spooler, and Desktop Organizer

    [F7]  Keyboard Enhancer, Spooler, Desktop
           Organizer, and Outliner
    [F8]  Keyboard Enhancer, Spooler, Desktop
           Organizer, and Thesaurus

    [F10] No Programs in the Background

Choose by Function Key:_
```

Figure 2: Menu screen display created by TSR.MNU

comparison, a version of TSR.MNU without the ANSI escape sequences appears in the Type entry. Entering the version in the Type entry first and then adding the escape sequences will ensure that all the text lines up properly. Be sure to enter the escape sequences in insert mode, not overtype mode.

TSR.MNU

```
←[0;59;"help";13p ←[0;60;"graphics";13p ←[0;61;"carousel";13p
←[0;62;"windows";13p ←[0;63;"key";13p ←[0;64;"group1";13p
←[0;65;"group2";13p ←[0;66;"group3";13p ←[0;68;"notsr";13p
←[0;34;42m←[2J←[1B
←[8C╔═══════════════════════════════════════════════════════════════════════╗
←[8C║ ←[1;37;41m                    CONFIGURATION MENU                       ←[0;34;42m ║
←[8C╟──────────────────────────────────────────────────────────────────────╢
←[8C║   ←[7m[F1]←[1;37;42m  Help with this menu                                ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F2]←[1;37;42m  DOS GRAPHICS Utility                               ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F3]←[1;37;42m  Software Carousel                                  ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F4]←[1;37;42m  Microsoft Windows                                  ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F5]←[1;37;42m  Keyboard Enhancer only                             ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F6]←[1;37;42m  Keyboard Enhancer, Spooler, and Desktop Organizer  ←[0;34;42m ║
←[8C║                                                                       ←[0;34;42m ║
←[8C║   ←[7m[F7]←[1;37;42m  Keyboard Enhancer, Spooler, Desktop                ←[0;34;42m ║
←[8C║         ←[1;37mOrganizer, and Outliner                                ←[0;34;42m ║
←[8C║   ←[7m[F8]←[1;37;42m  Keyboard Enhancer, Spooler, Desktop                ←[0;34;42m ║
←[8C║         ←[1;37mOrganizer, and Thesaurus                               ←[0;34;42m ║
←[8C║                                                                       ║
←[8C║   ←[7m[F10]←[1;37;42m No Programs in the Background                      ←[0;34;42m ║
←[8C╚═══════════════════════════════════════════════════════════════════════╝
```

The first three lines of the TSR.MNU file assign strings to the function keys. The function keys are represented by their extended ASCII codes (see ASCII code, Extended), a 0 followed by a semicolon and another number. Figure 2 depicts which string is assigned to which key. Each string ends with the carriage-return character 13 so that the batch file will be executed as soon as the key is pressed.

The fourth line establishes the colors (blue characters on a green background, established by ←[0;34;42m). It does not matter in which order the foreground and background parameters are entered. The 0 parameter (which resets all video attributes and is included to ensure that the

characters will be in normal intensity) must precede the colors or it will cancel them entirely.

On the same line, the sequence ←[2J clears the screen, and sequence ←[1B moves the cursor down one line. The latter sequence is not really necessary; in this instance, it would have been just as easy to insert a blank line in the text file. However, these cursor-moving sequences are quite useful for accurately positioning the cursor on a blank screen.

Each line of the file begins with the sequence ←[8C, which moves the cursor eight spaces to the right and thus centers the entire display. (You can also achieve the same result by entering eight spaces or a tab character (Ctrl-I).)

The rest of the escape sequences control color. The key caps appear in inverse video (black on light gray), set up by the sequence ←[7m. The title appears in bright white against a red background. In monochrome, the key caps would still appear in inverse video, and the title in highlighting. The remainder of the text is in bright white, and the frame is in normal-intensity blue. Notice that you must change the foreground color at the end of each line of text to accomplish this.

The above procedure is not the easiest way to create screens. However, it's the easiest way to create or control displays at the operating-system level, short of creating screens in a programming language and writing a program to display them.

Of course, once the batch files are executed, you probably won't want a prompt that says

```
Choose by Function Key:
```

Neither do you want the function keys to invoke these batch programs in the middle of an application or after the set of TSRs is loaded. Therefore, each of the batch files should contain the following lines:

```
prompt $p$g
←[0;59;0;59p ←[0;60;0;60p ←[0;61;0;61p ←[0;62;0;62p
←[0;63;0;63p ←[0;64;0;64p ←[0;65;0;65p ←[0;66;0;66p
←0;68;0;68p
```

This restores a standard drive\directory prompt and clears all the function key reassignments by assigning the keys to themselves.

Warnings There is no way to predict whether key reassignments will still be in effect when you load a given software package. Some programs take

over the keyboard, and others do not. If you have reassigned keys, you may find yourself entering incorrect commands when using your software if the program does not take over the keyboard. On the other hand, if you expect the reassignments to carry over and they do not, you may be equally stymied.

As a safety measure, always test any key reassignments with the program in which you want to use them before you open any vital files.

Key reassignments made via ANSI.SYS may not behave as expected (even at a DOS prompt) if you use either a keystroke-macro recording program or a similar program that changes the layout of your keyboard.

If you use a Prompt statement to change colors or to move the cursor in a batch file, your standard prompt will be replaced by a color scheme or a cursor movement. You must issue another Prompt statement with the correct characters for your prompt (see Prompt) after issuing such a sequence if you want your prompt to display information.

Tips

You can use the clear-screen sequence as a substitute for the CLS command. This practice is useful if you have a color graphics adapter, set a border color, and use a TSR that automatically clears the screen to black (see TSR). The clear-screen sequence does not clear the border, but the CLS command does.

Text that appears as underlined on a monochrome screen is blue on black on a color screen. The only color combination on a color screen that appears as inverse on a monochrome screen is black text on a white background.

You can use concealed text to issue a command that you don't want a user to see, or to suppress messages that result from the execution of a command. On a color screen, however, the concealed-text escape sequence results in black on black (unless the high-intensity video attribute is set, in which case the result is dark gray on black). To conceal text on a color screen, use the same color (with normal intensity) for foreground and background. If high intensity has been set, you can turn it off by including a 0 parameter before the rest of the parameters. For example, ←[0;35;45m will display magenta text on a magenta background.

If your software allows you to use ANSI key reassignments, the safest way to use reassigned keys is by creating a text file containing the reassignments you want to use as well as another text file that cancels them. Then call your program from a batch file that includes Type commands. You will be able to display both the file with the reassignments before you

invoke your program and the file that cancels them after you exit. Files shorter than one line display nothing on the screen; files longer than one line will display a blank line for each line of the file.

If you separate screen-control escape sequences with spaces, those spaces will appear in the display. Key-reassignment escape sequences needn't be separated by spaces. They are shown here with spaces only to enhance clarity.

Tables of ANSI escape sequences and parameters appear in Appendix B (for reference).

APPEND

Syntax

{1} append [*drive*]*path*;[*drive*]*path*; . . .
{2} append ;
{3} append /e
{4} set append=[*drive*]*path*;[*drive*]*path*; . . .
{5} set append=
{6} append /x
{7} append /x[:*setting*]
{8} append /path:*setting*

DOS Versions

{1}{2}	MS-DOS 3.2
{1}{2}{3}{4}{5}{6}	MS-DOS 3.3 to 3.31
{1}{2}{3}{4}{5}{6}	PC-DOS 3.3
{all}	MS-DOS 4.0
{all}	IBM DOS 4.0

Overview

The Append command establishes a search path for nonexecutable files. Append is used primarily with hard disks; if you use Append, you may find it easier to invoke software whose directory is not current.

A Path statement tells DOS to search all directories that are its parameters for executable files. However, it has no effect on files that are not executable. The Append command supplements the Path command in directing DOS to find files such as overlays, device drivers, and data files.

This allows you to execute programs that use such files without specifying their full path name, even if their directory is not current. In DOS 4.0, you can use the Append command to replace a path statement through judicious use of the command's switches.

Procedure Append is an external command; using it requires having the DOS program APPEND.COM (or APPEND.EXE) on a disk in your computer when you invoke the command for the first time. If the program is on neither the current directory nor the path, preface the command with a drive name and/or path name; for example,

```
c:\dos\append c:\;c:\dos;c:\batch
```

With every form of the Append command, enter the command followed by the path names of the directories that you want DOS to search for nonexecutable files. The path names must be complete path names, starting from the root directory, and beginning with a backslash (\).

When you invoke the Append command for the first time, it places a portion of the file APPEND.EXE (or APPEND.COM) in memory. This reduces available memory by about 1.7K in DOS 3.2, 3K in DOS 3.3 and later. Once this code is in memory, you can invoke Append as though it were an internal command; it need not be available on a disk, and it need not be prefaced with a drive name or path name.

You may include or omit the drive letter for any of the directories. If you include the drive letter, DOS searches the specified drive, regardless of which drive is current. Each path name must be separated from the next by a semicolon; do not include any spaces between the path names.

You needn't duplicate the parameter of the Path command. For example, you probably want your DOS directory on the search path established by the Path command. However, all the DOS files (except for device drivers) are executable commands, and you would gain nothing by making the DOS directory a parameter of the Append command. (It won't help with the device drivers because they are loaded by the CONFIG.SYS file, which DOS reads before it can execute any Append statement.) However, if you use the DOS 4.0 DOSShell program, you should include the DOS directory as a parameter to the Append command, because it uses several data files that are in that directory.

Normally, the Append command is placed in an AUTOEXEC.BAT file. With DOS 4.0, however, you can use the Install command to load Append into memory from your CONFIG.SYS file. (See CONFIG.SYS File and Install = .) If

you do, you will not be able to boot your computer without the Append command unless you rewrite your CONFIG.SYS file to exclude this command.

You can change the list of appended directories at any time by issuing a new Append statement, which will replace the previous list.

The Standard Append Command

{1} append [*drive*]*path*;[*drive*]*path*; . . .
{2} append ;

The first version of the Append command is available both in MS-DOS 3.2 and 3.3, and PC-DOS 3.3. To see which directories are arguments to the Append command, enter append with no parameters. To delete the arguments, enter append ;

Placing the Append List in the Environment

{3} append /e
{4} set append=[*drive*]*path*;[*drive*]*path*; . . .
{5} set append=

An alternative method of establishing a search path for nonexecutable files is available in PC-DOS 3.3, IBM DOS 4.0, and MS-DOS 3.3 to 4.0. This method requires two Append statements, and places the list of appended directories in the environment. First you must issue a command of the form

[*drive*][*path*]append /e

The /e switch tells DOS to place the list in the environment. You can also include the /x switch, described below, if you wish. Once you have done this, you then issue a second Append statement which includes the list of appended directories. This Append statement can take the standard form already described, or it can use the Set command:

set append=[*drive*]*path*;[*drive*]*path*; . . .

To delete the arguments, you can enter either

append ;

or

set append=

To see the argument list, you can enter either append or set. The Set command also displays other information, including the path, the prompt, and the default command processor.

If you want to place the list in the environment, you must have enough environment space to hold the names of all the directories you want to include. If you do not, you will see the message

```
Out of environment space.
```

You cannot place the list in the environment once an Append statement with a list of directories has been issued. The /e switch works only the first time you invoke the Append command.

See Shell=, Command, and Environment Space for information on how to change the size of the environment. See Set for more information on the Set command.

Extending the Scope of the Append Command

{7} `append /x`

The /x switch extends the Append command so that it works in other circumstances as well. When you use the /x switch, DOS commands that locate files (for example, Dir, Copy, Rename, and Erase) search appended directories for file names that match the one you specify. The search, however, extends only to the first directory containing a matching file name. A *.* wildcard does not extend the search at all.

Like the /e switch, the /x switch must be used the first time you enter an Append statement. You may use either or both in this first statement, but you should not include a list of appended directories.

Extending the Search in DOS 4.0

{7} `append /x[:setting]`
{8} `append /path:setting`

DOS 4.0 includes several new capabilities in its Append command. First, although you must use the /e switch before appending specific directories, you can now use the /x switch at any time, even after directories have been appended. Moreover, the /x switch has a different effect. When you use the /x switch in DOS 4.0, it extends the search for executable files as well as nonexecutable files. (However, it doesn't affect file names that are the parameters to commands, as the /x switch in earlier releases does.) Thus,

Append /x can substitute for the Path statement. Used in this manner, the Append command can reduce the amount of environment space you need, because you don't need to install a Path statement in the environment.

To turn this capability off and on once you have placed Append in memory, you can use the commands append /x:off and append /x:on. The latter is the same as append /x. However, if you change the list of appended directories before you first turn this extended capability on, the list will revert to the first list you loaded. To avoid this problem, use the /x switch in the same command as your new list of appended directories, after the list; for example,

```
append c:\;c:dos;c:\batch;c:\wp /x:on
```

You can further extend the way Append searches for executable files with the /path parameter. Like the /x parameter, it can be invoked at any time. When you enter the command append /path or append /path:on, DOS will search all appended directories for an executable file, even if you include a drive and/or path name in your command. Thus for example, if you issued the command c:\lotus\123 and 123.COM was on a directory in the appended list, but not in C:\LOTUS, DOS would find and execute it anyway.

If you don't specify these switches, the default condition is for /x to be off and /path to be on. You can turn off Append's ability to override specified drives and directories by entering the command append /path:off.

Examples

Without the Append command, some software will not load properly unless the directory in which it resides is current, even if the directory is on the search path. This is because neither the software nor DOS knows where to look for nonexecutable files such as overlays, configuration files, and so on. Moreover, it is often useful to keep data files in directories separate from your software. If you do, you will want to call your programs from the directory where your data files are located, rather than from the program's directory.

Lotus 1-2-3 is one program that normally must be loaded from its home directory, even if that directory is on the search path. If you call 1-2-3 with a command such as c:\lotus\123 the program will not load if the current directory is other than the one in which Lotus 1-2-3 is located. Even if there is a valid device driver set on the key disk, a message will appear

saying that the device driver set cannot be found. With the C:\LOTUS directory appended, however, you can call 1-2-3 from anywhere in your system. (Of course, you still need the key disk.)

Similarly, SideKick and SuperKey can neither find their help files nor accept modifications to their configuration unless their directory is current when you call them. With their directories appended, on the other hand, you can call or modify them from anywhere. If you use these three programs, the statement

```
append c:\lotus;c:\sidekick;c:\superkey
```

should appear in your AUTOEXEC.BAT file.

Warnings

Commands that respond to the /x switch in DOS 3.3 behave capriciously, and the consequences can be disastrous. Suppose, for example, that you are looking for a file with the extension .DOC. There is none on the current directory (which is C:\REPORTS), but there are several on appended directories. If you use the command `dir *.doc` to look for the file, the result will be a directory display similar to Figure 3. However, none of these files is actually on the C:\REPORTS directory. In fact, they are all on the first appended directory that has any files with a .DOC extension. If you call your word processor, you will probably be able to open these files from the C:\REPORTS directory. Unfortunately, it is difficult to know where they will be written: they may be written to the C:\REPORTS directory, or to their original directory—or your word processor may report a disk error and not write them at all.

```
C:\REPORTS>dir *.doc

 Volume in drive C is MY SYSTEM
 Directory of  C:\REPORTS

SALES_8   DOC     56160   8-12-88    1:23p
BERKEY    DOC      1034   8-16-88    3:06p
SALES_7   DOC      1164   7-08-88    8:42a
KERMIT    DOC       716   8-16-88    2:12p
GONZO     DOC     21312   6-10-88    9:23a
SALES_6   DOC     42768   6-11-88    1:05p
PIGGY     DOC      4480   8-17-88    4:02p
BIGBIRD   DOC     35840   8-18-88    2:30p
SALES_5   DOC     68448   5-05-88    3:10p
        11 File(s)   9650176 bytes free
```

Figure 3: Directory display with Append /x in use

Similarly, the /x switch in DOS 3.3 can cause problems if you copy files. For example, if you see the above directory, you might inadvertently overwrite a file with an older version from another directory.

Tips

If you place the appended list in the environment, you can add directories to it without retyping the entire command. Create the batch file SRCHMORE.BAT, containing the line

```
set append=%append%;%1
```

To add a directory called C:\DATA to the appended list, enter the command

```
srchmore c:\data
```

However, if the entire Append statement becomes longer than 132 characters, you will get a `Syntax error` message.

ASCII Code

Overview

ASCII (American Standard Code for Information Interchange) is a code that assigns a number to the letters, numbers, and punctuation marks on your keyboard and to other characters that are not on the keyboard. Your computer actually sends a binary number representing the ASCII code of a character each time it sends a character to the screen, a printer, a modem, or other peripheral device. Depending on the purpose, these codes are represented in either decimal or hexadecimal notation. Although MS-DOS includes codes for characters numbered 0 through 255, only the first 128 (codes 0 through 127) are completely standardized. For a complete ASCII table, see Appendix B.

ASCII Codes 0 through 31 represent *control characters*, which are produced by pressing a combination of the Ctrl key and another key. These key combinations generally issue commands, either to hardware or software. Control keys all have a second name that indicates their device-control effect. Thus, Ctrl-G is BEL, Ctrl-[is ESC, Ctrl-M is CR (Carriage Return), and Ctrl-I is HT (Horizontal Tab).

DOS also assigns graphic symbols to these codes. Some programs display the symbols associated with some or all of the control characters on the screen; others do not.

Control characters are represented on your computer either by a caret mark (^) followed by the character designation, or by its graphic equivalent.

For example, ^G is Ctrl-G, which sounds a beep; it may also be represented by a small diamond. ^Z is Ctrl-Z, the end-of-file mark (ASCII code 26); it may also be represented by an arrow pointing to the right. ^[is the escape character (ASCII code 27); it may also be represented by an arrow pointing to the left. In this book, it is represented by the arrow.

If you use the Type command to display a file that contains control characters, some will be represented by their graphic equivalents, and others will have their normal device-control effect. Among the latter are Ctrl-M (which issues a carriage return), Ctrl-I (which issues a tab character), and Ctrl-G. When a Ctrl-M is encountered, the cursor moves to the beginning of the next line. When a Ctrl-I is detected, the cursor moves to the next screen column that is a multiple of 8. You will get the same effect if you enter these characters on the command line by holding down Ctrl and pressing the key named in the code.

ASCII code 32, the space character, is produced by pressing the space bar. Codes 33 through 47, 58 through 64, 92 through 96, and 123 through 126 represent the nonalphanumeric characters (such as punctuation marks) on the keyboard. Numerals are represented by codes 48 through 57, the uppercase letters by 65 through 91, and the lowercase letters by 97 through 122. ASCII 127 is the delete character.

The codes from 128 through 255 represent foreign language and graphics characters. The graphics characters are often used to create borders in screen displays, and the foreign-language characters and currency symbols are used by the various commands and files that prepare the computer to use different languages. See ChCP, Country=, DISPLAY.SYS, GrafTabl, KEYB, Keyb*nn*, KEYBOARD.SYS, and NLSFunc for more information.

Procedure/ Examples

Various programs and devices may require you to send characters to them as their ASCII values. For example, you can enter a text character in DEBUG (see DEBUG) by entering its hexadecimal ASCII code. You need to enter the decimal ASCII codes of characters whose keys you want to reassign using ANSI escape sequences (see ANSI.SYS).

Printers are controlled by escape sequences. Depending on how you send the escape sequences to the printer, you may need to use either the decimal or hexadecimal ASCII code for the characters in the sequence. Any of the following three methods can be used to send escape sequences to your printer.

Assume that the Esc Ctrl-O code sets your printer to condensed type. If you include the characters in a batch file called CONDENSD.BAT, the file will contain the line

```
Echo ←^O >prn
```

You cannot issue this command on the command line because the command line responds to the Escape key as a command; however, you can create it in EDLIN (see EDLIN). If you create a BASIC program to do the same thing, the program will contain the line

```
10 LPRINT CHR$(27);CHR$(15): SYSTEM
```

using the decimal ASCII values. If you program in assembly language, you can write a program sending the same codes to the appropriate port using their hexadecimal representations. Your printer manual will give you additional information.

See Graphics Characters, Displaying and Printing, for information about entering the characters with codes above 128 and about using graphic representations of control characters.

ASCII Code, Extended

Overview The extended ASCII code is a set of codes for keys that do not have standard ASCII codes. These keys include:

- The function keys
- Shifted function keys
- Ctrl-function keys
- Alt-function keys
- Other Alt-key combinations
- The cursor keypad keys
- The separate numeric-keypad keys on the extended (101-key) keyboard

To distinguish these codes from standard ASCII codes, each numeric code is preceded by a zero and a semicolon. Thus, the code for Alt-X is `0;45`.

A complete list of extended ASCII codes appears in Appendix B. Note that the order in which characters appear corresponds to their positions

on the keyboard, rather than to the positions of their unaltered equivalents in the standard ASCII code. Note also that the table of extended ASCII codes in Chapter 6 of most editions of the *PC-DOS Technical Reference* is incorrect.

Procedure/ Examples

If you are not a programmer, the most common use for the extended ASCII codes is reassigning keys using the ANSI.SYS device driver (see ANSI.SYS). For example, if you wanted to have the F10 key issue the command `dir /p`, you would use the Type command to display a text file containing the characters

```
←[0;68;"dir /p";13p
```

The first part of this escape sequence, `0;68`, is the extended ASCII code for the F10 key; the text `dir /p` is the text you want to display; 13 is the normal ASCII code for a carriage return; and the semicolons separate the elements of the escape sequence.

To clear this assignment, display a text file containing

```
←[0;68;0;68p
```

This reassigns F10 to F10.

Assign

Syntax

`assign` *`virtualdrive=physicaldrive`*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Assign command gives a new letter designation to a disk drive; it does not change the old drive name. Therefore, when you use the Assign command, the affected drive will have two names. The new name must be the same as a real disk drive in your system, so you won't be able to use the real drive with the same name until you cancel the assignment.

ASSIGN

A

The command allows you to use software that expects certain files to be on drives with specific names even though you do not have such drives in your system.

Procedure

Assign is an external command; using it requires having the DOS program ASSIGN.COM (or ASSIGN.EXE) on a disk in your computer. If you use DOS 2, it must be either on the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:assign b=a
```

If you use DOS 3.0 or later and the program is not on the current directory or the path, preface the command with a drive name and/or path name; for example,

```
c:\dos\assign b=a
```

Programs that are designed specifically for one-drive or two-drive systems let you indicate neither the drive and directory where they are located nor the drive and directory where you keep their data files. Instead, these programs have specific drive names written into the program code. The Assign command allows such programs to run on systems where drive names do not match those embedded in the code.

To cancel a drive reassignment, type `assign` at a DOS prompt, with no parameters.

Examples

Imagine that you're using a borrowed copy of WordPerfect that was installed for a two-diskette system. However, your system has one diskette drive and one hard disk. The program is installed so that the program diskette is to remain in drive A, and the data files are to be created on drive B. You, however, have drives named A and C, and want your data files to be on drive C, your hard disk. If you enter the command

```
assign b=c
```

your computer will behave as though the hard disk were drive B as well as drive C. You can then use the program. However, the commands `dir b:` and `dir c:` both produce the same display. Only the first line of the display, beginning `Volume in Drive` *d* . . . would be different. The drive name would be the one you entered.

Warnings

You cannot create a new logical drive with the Assign command; Assign can only assign one physical drive or RAM disk to another. In other words, both drives in an Assign statement must already exist in the computer.

The Assign command is very dangerous. If a drive reassignment is in effect, the Backup, ChkDsk, Restore, Join, Subst, and Print commands will not behave as expected. (Format and DiskCopy are not affected.) You may run into trouble if you use TSRs that are accessing a drive you want to reassign. Moreover, if your Assign statement reassigns the disk on which the ASSIGN.EXE or ASSIGN.COM program resides, the command becomes unavailable. Therefore, if the program were on drive A in the example above, you would need to reboot to undo the reassignment.

While a reassignment is in effect, it is easy to become confused about which drives you can still access. As a result, you might inadvertently erase the wrong iteration of a file or perhaps copy an older version of a file over a newer one when you intended the reverse. Because of the potential problems associated with this command, try to avoid it.

Tips

DOS versions 3.1 and later contain two new commands, Join and Subst (see Join and Subst), which are both more powerful and safer than the Assign command. Join allows you to treat a drive as a directory of another drive, and Subst allows you to treat a directory as though it were a separate drive.

If a set of reassignments makes a physical drive unavailable, include a reassignment that gives a new name to the physical drive. In the example above, the command

```
assign a=c b=c d=a
```

allows you to access drive A by referring to it as drive D (assuming you have one). However, if you use DOS 3.0 or later; and need to refer to drives with letters after E, you must include a `lastdrive=` statement in your CONFIG.SYS file (see CONFIG.SYS File).

In some versions of DOS, you can make successive reassignments with commands such as

```
assign b=a
assign a=c
```

In those DOS versions, these two commands have the same effect that

```
assign b=a a=c
```

has in more standard versions. In most versions, one Assign statement cancels previous Assign statements. To maintain an assignment such as `assign b=a` after issuing another assignment, you must include both assignments in the second Assign statement. For alternatives, see Join and Subst.

Attrib

Syntax

{**1**} attrib ±r [*drive*][*path*\]*filename.ext*
{**2**} attrib ±a [*drive*][*path*\]*filename.ext*
{**3**} attrib [*drive*][*path*\]*filename.ext*
{**4**} attrib [*drive*][*path*\]*filename.ext* /s

DOS Versions

{**1**}{**3**}	MS-DOS 3.0 to 3.1
{**1**}{**3**}	PC-DOS 3.0 to 3.1
{**1**}{**2**}{**3**}	MS-DOS 3.2
{**1**}{**2**}{**3**}	PC-DOS 3.2
{**all**}	MS-DOS 3.3 to 4.0
{**all**}	PC-DOS 3.3
{**all**}	IBM DOS 4.0

Overview

The Attrib command allows you to examine or change some of a file's *attributes* (see File Attributes). The Attrib command in DOS releases 3.0 to 3.1 can address only the read-only attribute; the release 3.2 to 3.3 versions can address the archive attribute as well. (No version, however, can address the hidden or system attributes.) Attrib can address a file that is not on the current drive and directory if you preface the file name with a drive name and/or path name. It will also accept file names with wildcard characters. In DOS 3.3 and later, it can address files in subdirectories of the current, or the specified, directory, if you add the /s switch.

Procedure

Attrib is an external command; using it requires having the DOS program ATTRIB.COM (or ATTRIB.EXE) on a disk in your computer. If the program

is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\attrib -r filename.ext
```

Attrib is one of the few DOS commands that uses a switch before the file name parameters. A switch with a plus sign activates, or sets, the attribute; a switch with a minus sign deactivates, or clears, the attribute.

A file attribute is established by setting or clearing bits in the *attribute byte* in the file's directory listing. File attributes do not normally appear in directory listings. However, some utility programs can display all of a file's attributes, and the Attrib command can display those attributes that it can address (see File Attributes).

Addressing the Read-Only Attribute

{1} `attrib ±r [drive][path\]filename.ext`

The /r switch stands for *read-only*. When a file has its read-only attribute set, you cannot modify its contents, erase it, or replace it by a new file of the same name. You can set the read-only attribute of any file you want to protect by issuing a command of the form

```
attrib +r [drive][path\]filename.ext
```

The read-only attribute has three separate effects, and the command that changes it is useful in several situations. For example, read-only files cannot be modified, so you can use this attribute to protect files that are shared by many users. Also, since read-only files cannot be erased, you can use this attribute to prevent files from being erased accidentally. Finally, since they cannot be replaced, you can prevent a file from being replaced inadvertently by an older version.

If you later need to modify a file that has been marked read-only, you can clear the read-only attribute by issuing a command of the form

```
attrib -r [drive][path\]filename.ext
```

Displaying File Attributes

{3} `attrib [drive][path\]filename.ext`

Entering the Attrib command with no switches displays a directory of all files matching the specified file name. An R will appear in the leftmost screen column next to the names of read-only files, and an A will appear in the third screen column next to the names of files whose archive

attribute is set. (See Figure 4 for an example.) The Attrib command displays only file names that appear in a normal directory listing; thus, it does not display the names of hidden or system files.

Addressing the Archive Attribute

{2} `attrib ±a [drive][path\]filename.ext`

The /a switch stands for *archive*. You can use the archive attribute to determine which files will be copied (archived) by the Backup and Xcopy commands. The archive attribute is a signal that indicates whether a file is new or has been changed since it was last backed up. The archive attribute is set whenever a file is created or changed, and cleared when it is archived.

When you use the Backup command with the /m switch, or Xcopy with the /a or /m switch, only files whose archive attribute is set are backed up or copied. With the /m switch, these programs clear the archive attribute after copying a file. Therefore, the file will not be copied again when the same command is used unless it has been modified since the previous backup. This feature allows you to make *incremental* backups of your files. As a result, you needn't back up every file on a disk—only those that have changed since your previous backup. (Note that the Copy command does not affect the archive attribute of the source of the copy. However, the new copy will have the archive attribute set.)

When you use the /m switch to Restore (available in MS-DOS 3.2 to 3.3 and DOS 4.0), a file will be restored only if the copy on the target disk has the

```
C:\> attrib *.*
  A      C:\COMMAND.COM
         C:\CONFIG.SYS
         C:\ANSI.SYS
         C:\ANSI43.SYS
         C:\DM.SYS
         C:\DRIVER.SYS
         C:\MSMOUSE.SYS
         C:\RAMDRIVE.SYS
         C:\EMM.SYS
  A      C:\AUTOEXEC.BAT
         C:\CAROUSEL.OPT
  A      C:\LASTBKUP
R        C:\MIRROR.FIL
R        C:\MIRORSAV.FIL
         C:\SPOOL.COM
         C:\TREEINFO.NCD
[cd]
```

Figure 4: Display created by the Attrib command

archive attribute set. Thus, this switch enables you to restore files to the state they were in before they were modified. (See Backup, Restore, and Xcopy.)

Repeating an Attrib Command on Subdirectories

{4} `attrib [drive][path\]filename.ext /s`

In DOS 3.3 and later, you can view or change file attributes in subdirectories of the current or the specified directory, as well as in that directory. To do so, add the /s switch (for subdirectories) at the end of the command.

Examples

Suppose you want to back up the modified files on your hard disk, but your word processor creates a backup copy of each file you edit (you can recognize these files by the extension .BAK). To avoid making backup copies of files that are themselves backup copies, issue the following command before beginning your backup procedure:

```
attrib -a \wp\*.bak
```

Assuming your word processing files are on the \WP directory, this command will clear the archive attribute from any files having the .BAK extension, and they will be skipped when you back up your modified files. If you keep word processing files on several directories, make a batch file with one such command for each directory. If you have many directories with word processing files, you might begin by issuing the command

```
tree>catalog.hd
```

This command creates a file called CATALOG.HD, which includes, among other things, all the directory names. (See Tree and Redirection and Pipes.) You can create the batch file by editing this file in your word processor, in nondocument or ASCII mode.

Suppose you want to prevent other users from changing the root directory of your hard disk, but that you also want to be able to modify your AUTOEXEC.BAT and CONFIG.SYS files (see AUTOEXEC.BAT File and CONFIG.SYS File). The following set of commands will have the desired effect:

```
attrib +r \*.*
attrib -r \AUTOEXEC.BAT
attrib -r \CONFIG.SYS
```

The first command makes all the files on the root directory of the hard disk read-only. The second and third commands clear the read-only attribute from the AUTOEXEC.BAT and CONFIG.SYS files. You need the second and third commands because the first command will affect these files along with all others in your root directory.

You might want to issue the first command to protect the latter two files as well, and clear the read-only attribute only when you explicitly want to edit the files. This strategy will protect these files along with all the other files in the directory.

Tips

You can address both attributes in the same command. A command of the form

```
attrib -a +r filename.ext
```

is perfectly acceptable. This command makes the specified file into a read-only file, and clears the archive bit so that it will not be backed up.

AUTOEXEC.BAT File

DOS Versions

MS-DOS 2.1 to 3.31
PC-DOS 2.0 to 3.31

Overview

The AUTOEXEC.BAT file is a special batch file (see Batch Files). When you boot your computer, DOS looks for this file on the root directory of the boot disk. If a batch file by this name is present, DOS executes it as soon as COMMAND.COM is in memory.

The AUTOEXEC.BAT file takes care of anything that you do (or should do) routinely when you start your computer. These tasks vary depending on several factors:

- Whether you boot from a hard disk or a diskette
- Whether you use a single application or many applications
- Whether you need to configure your hardware in ways other than those controlled by the CONFIG.SYS file

- Whether you use a RAM disk (see VDISK.SYS and RAMDRIVE.SYS)
- Whether you use TSR utilities

Generally speaking, the AUTOEXEC.BAT file sets up your system to take advantage of your hardware, establish a suitable environment for your applications, and conform to your needs. Several AUTOEXEC.BAT files for different situations appear in the Examples section.

Procedure

Because the AUTOEXEC.BAT file is a batch file and has the characteristics of any other batch file, it must be an ASCII text file. Thus, you can include in it any command that is executable from a DOS prompt, as well as the special batch file commands. The only other restriction is that AUTOEXEC.BAT file must be on the root directory of the boot disk.

Because the AUTOEXEC.BAT file is always executed automatically, some commands are more appropriately included in it than others. Be sure to include in your AUTOEXEC.BAT file only commands that you want to execute every time you boot your computer.

One effect of using an AUTOEXEC.BAT file is that DOS does not prompt you to enter the date and time. Therefore, you should include some way of telling your computer the correct date and time. If you don't have a real-time clock, include the Date and Time commands; if you do have a real-time clock, substitute the command that calls its driver. If you use a PC AT or compatible or a 386 PC, you can omit this step since DOS reads the time automatically from the CMOS memory chip.

Examples

A Diskette-Based System If you work entirely with diskettes, you should have an AUTOEXEC.BAT file on the startup disk for every application you use. (See Sys.) This allows you to tailor your system so that it provides the most suitable environment for each application. Of course, this means that you may have to reboot your computer when you start each application.

Let's say you use dBASE III PLUS with SideKick (a set of desktop utilities) and SmartKey (a keyboard macro program). Your AUTOEXEC file can read all the necessary files into memory before starting your application. The following program illustrates an AUTOEXEC.BAT file that takes care of

everything you need to do before you start working in your application. You will, of course, need to place the following files on your startup disk:

SKBATCH.COM (the file that lets you address SmartKey from a batch file)
SMARTKEY.COM
DBASE.DTX (SmartKey's keyboard definition file for dBASE)
SKC.COM (the version of SideKick without an appointment calendar)

You may have to configure this version of SideKick to find its data files on drive B. If you do, make sure they are on your data disk.

AUTOEXEC.BAT File for dBASE III PLUS with Diskettes

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`date`	Prompts you for the date
`time`	Prompts you for the time
`path a:\;b:\`	Tells DOS to look on drives A and B for executable files
`skbatch /l dbase`	Loads SmartKey and its macro file for dBASE
`skc`	Loads SideKick without the calendar
`dbase`	Invokes dBASE III PLUS

Now let's say you switch to WordPerfect for word processing. You don't need a keyboard macro program with WordPerfect because it has its own macros, but you can't unload SmartKey without unloading SideKick first, and you want to use SideKick. The easiest way to solve this is to reboot the computer. The AUTOEXEC file on your WordPerfect startup diskette would resemble the file shown in the following program. As with dBASE, all the files needed by the batch file must be on the startup disk. In this instance, you need add only SK.COM, the full version of SideKick. However, if you modified the SideKick data files on the dBASE data diskette, you will want to copy them to your WordPerfect data diskette before rebooting.

AUTOEXEC.BAT for WordPerfect with Diskettes

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`date`	Prompts you for the date
`time`	Prompts you for the time
`sk`	Loads the complete SideKick
`wp`	Starts WordPerfect

You don't need a Path command in this version because you can configure WordPerfect to find its own files on drive A and data files on drive B.

You needn't end the batch file with the command that starts your application. You can include commands to be executed after you exit the application as well. These commands might back up your revised data files to another disk or directory, clear the screen, show a directory, or anything else you find useful.

A Diskette-Based System with a RAM Disk A RAM disk can enable your AUTOEXEC.BAT file to execute more quickly, and your AUTOEXEC.BAT file can do a great deal to make your RAM disk useful. (See VDISK.SYS, RAMDRIVE.SYS.) To understand the advantages of a RAM disk, let's compare executing a batch file from DOS and from a RAM disk. When DOS executes a batch file, it executes one command at a time. After it executes each command, DOS goes back to the batch file to read the next command. This process, disk I/O, is the slowest computer function because it requires moving a physical object. In contrast, other computer processes merely move electrical impulses. A RAM disk, because it comprises only electronic storage space, can be read as quickly as any other portion of memory. Consequently, a batch file executes several hundred times faster from a RAM disk than from a diskette.

You can take advantage of this speed if you use a complicated AUTOEXEC.BAT file. Break the AUTOEXEC file into two parts, and call the first part AUTOEXEC.BAT and the second RAMDISK.BAT. If your RAM disk is drive C, your AUTOEXEC file will contain only the lines shown in the following program.

AUTOEXEC Code for a Diskette and a RAM Disk

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`copy ramdisk.bat c:`	Copies RAMDISK.BAT to the RAM disk
`c:`	Makes the RAM disk current
`ramdisk`	Executes RAMDISK.BAT

These lines place RAMDISK.BAT on the RAM disk, make the RAM disk the default disk, and execute RAMDISK.BAT. The latter file contains everything else that you would normally place in your AUTOEXEC file.

Let's consider what should go into RAMDISK.BAT. Besides whatever else your AUTOEXEC.BAT file does, it can also do the following:

- Force DOS to find COMMAND.COM on your RAM disk
- Execute itself from the RAM disk
- Transfer batch files (or other programs) that you use often to your RAM disk

If DOS can find COMMAND.COM on your RAM disk, you should never again see the message

```
Insert COMMAND.COM disk in drive A:
Then press any key when ready
```

You already know the advantage of executing most or all of AUTOEXEC.BAT from the RAM disk. Placing other batch files on your RAM disk is advantageous for the same reason: the files execute more quickly. You may also want to include such utilities as FORMAT.COM, CHKDSK.COM, and DISKCOPY.COM just to keep them available in your computer. Alternatively, if you use resident utility programs, you can keep their help files or reference files on the RAM disk (see TSRs). Hence, they will always be on hand when you need them, even if you change diskettes.

With this kind of procedure, you should use two diskettes to start your system since you probably won't have enough room on the boot disk for everything needed to establish your working environment and start

your application. Therefore, begin with a disk just to establish your environment, then switch to your application disk. Using two diskettes is a small price to pay for the convenience of having the additional files always available.

The way to place COMMAND.COM on the RAM disk depends on your version of DOS. If you use DOS 3.0 or later, use the procedure already explained to execute the file. The first lines of RAMDISK.BAT will be

```
copy a:command.com>nul
set comspec=c:\command.com
```

These lines copy COMMAND.COM to the RAM disk and tell DOS to look for it there. (See Set.)

These same lines won't work with DOS 2. If you use the Set command to look at the environment, you'll see the statement

```
COMSPEC=C:\COMMAND.COM
```

This may lead you to think you have accomplished your goal, but DOS 2 won't pay attention to your set comspec statement. Instead, you need to load a *secondary command processor* (see Command). To do this and also have DOS execute your AUTOEXEC.BAT file requires some tricky coding—in this case, a variation of the previous program as well as some special coding in RAMDISK.BAT. The following program shows the code for AUTOEXEC.BAT.

AUTOEXEC Code to Place COMMAND.COM on a RAM Disk Using DOS 2

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`c:`	Makes the RAM disk current
`copy a:command.com>nul`	Copies COMMAND.COM to the RAM disk
`copy a:ramdisk.bat c:autoexec.bat>nul`	Copies RAMDISK.BAT to the RAM disk, renaming it AUTOEXEC.BAT at the same time
`command c:\ /p`	Loads COMMAND.COM from the RAM disk

This strange bit of code starts by making the RAM disk current, then copies COMMAND.COM and RAMDISK.BAT to the RAM disk, renaming the latter to AUTOEXEC.BAT. The trick is in the line

```
command c:\ /p
```

This command loads a second copy of COMMAND.COM into memory from the RAM disk. The drive and directory name, `c:\`, tell DOS where to read COMMAND.COM from. Finally, the /p switch tells DOS to make this a permanent copy.

As soon as COMMAND.COM is loaded, it looks for an AUTOEXEC.BAT file to execute. However, as far as COMMAND.COM is concerned, drive C has become the boot disk. Since an AUTOEXEC.BAT file is on the boot disk, COMMAND.COM executes it.

When a new command processor is loaded, it creates its own environment. Therefore, the environment associated with the previous command processor disappears. In practical terms, this means that if you don't want the batch file to display its commands on the screen as they are executed, you must begin the RAMDISK.BAT file with the command `echo off`. When you chain batch files, echo usually stays off if you turned echo off in the first batch file (see Echo).

Remember to include the Date, Time, Path, and Prompt commands in RAMDISK.BAT. You can also include commands to set up your system in whatever way you prefer.

A Hard-Disk–Based System When your system includes a bootable hard disk, an AUTOEXEC.BAT file is especially important and deserves careful consideration. You can use this file quite effectively to customize your working environment, but you may not want to have it start applications or even to load resident utility programs (see TSRs).

If you use a hard-disk system, you probably use more than one application program. If so, you don't necessarily want your computer's environment to be the same for all of the application programs. Under these circumstances, the AUTOEXEC.BAT file should just take care of the following housekeeping chores:

- Setting the date and time
- Establishing a default path, prompt, and append list

- Setting up your RAM disk, if you use one
- Backing up the new and changed files from your hard disk

You have already learned about setting the date and time. The additional steps are specific to hard-disk systems.

With a hard disk, you should always establish a default path which should include at least the following:

```
path c:\;c:\dos;c:\batch;c:\util
```

You may also want to include several software directories. However, if the \BATCH directory is already on your path, you can always use your batch files to change the path as needed. (See Batch Files, Directories, Path, and Append.)

Since your hard disk will have subdirectories, you should modify your prompt to show, at minimum, the current directory as well as the current disk. To accomplish this, use the command

```
prompt $p$g
```

See Prompt and ANSI.SYS for details on ways to add other information to your prompt.

If you've never had a hard-disk failure, it's easy to forget to back up your hard disk daily. However, when you do experience a failure (and it certainly will happen at some point), you'll be glad you have done so. For code that ensures that you back up your hard disk once every day, see Hard Disks, Backing Up and Restoring.

Many people like to have their AUTOEXEC.BAT file load their resident utilities (see TSRs). This is not always a good idea. If you use memory-hungry programs such as AutoCAD, Ventura Publisher, or Symphony, you should leave as much system memory free as you can. One way to give yourself a choice is to include lines such as the following in your AUTOEXEC.BAT file, before the lines that load the TSRs:

```
echo Resident utility programs will be loaded next.
echo To skip this step, press Ctrl-Break. Otherwise,
pause
```

The Pause command displays the message

```
Strike a key when ready . . .
```

and waits for a keypress, which gives you a chance to break out of the batch file before the TSRs are loaded (see Pause).

Another technique is to follow the housekeeping commands with a call to something like the TSR menu program presented in the ANSI.SYS entry.

Given these considerations, if you use a hard disk your AUTOEXEC.BAT file will resemble the following program.

AUTOEXEC.BAT for a Hard-Disk–Based System

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`clock`	Reads the time from the clock
`path c:\;c:\dos;c:\batch;c:\util;c;\`*`app1`*`;c:\`*`app2`*	Sets a default search path including two directories containing application programs
`append c:\util;c:\`*`app1`*`;c:\`*`app2`*	Establishes a list of appended directories, including the \UTIL directory and the two application directories that are on the search path
`prompt $p$g`	Sets the prompt to show the default drive and directory
`command /c hdbackup`	Branches to the hard-disk backup batch file
`tsr`	Calls the TSR menu program

If you use DOS 3.3 or later, you can substitute the command `call hdbackup` for `command /c hdbackup`. See Call and Command for further information about these commands.

The reasons for running a RAM disk with a hard disk are somewhat different from running one with a diskette-based system. First, hard disks are much faster than diskettes, so you don't save quite as much time. Second, COMMAND.COM, your DOS programs, and most if not all the utilities you need are permanently in your computer, on your hard disk. (If they aren't, they should be.) However, you may still find a RAM disk helpful with certain types of programs. For example, batch files execute

considerably faster from a RAM disk than from a hard disk. If you frequently use complex batch files, you will notice a definite gain in speed if you execute your batch files from a RAM disk.

Programs that read from disk frequently also benefit from a RAM disk. You can save even more time if your applications that use overlay files can find them on the RAM disk.

To accomplish these ends, include commands to copy the appropriate files to the RAM disk in your AUTOEXEC.BAT file. You might even set up a special directory just for files that you want on the RAM disk, then copy them all at once with the single command

```
xcopy c:\ramdisk d:
```

or, if you use DOS 3.1 or earlier,

```
copy c:\ramdisk\*.* d:
```

You can take the process even further by having two RAM disks (a small one for your batch and overlay files, and a second, larger one for data files). If you do so, add the small one to your search path. You may have no trouble accessing the overlays if you also add the root directory of this disk to your Append list (see Append).

Special Hardware Considerations Regardless of what kinds of disks you have, you must load special programs to run some types of hardware such as serial printers, monochrome graphics adapters, and multiple monitors. You should always include calls to these programs in your AUTOEXEC.BAT file.

If you use a serial printer, you must tell DOS how to communicate with it. This requires two commands; the first sets up the serial port so that it sends out the information in the form your printer expects to receive it, and the second redirects the printer output to a serial port. You effect this through two calls to the Mode command:

```
mode com1:bps,parity,databits,stopbits,p
mode lpt1:=com1:
```

The first of these lines configures the first serial port. (Substitute COM2 if your printer is attached to that port.) The most common settings are 12,n,8,1. Check your printer manual to determine the correct settings.

The second command directs the printer output to the serial port. (See Mode.)

If you use both a monochrome and a color monitor, you can control which monitor will be active when you start your computer. Include in your AUTOEXEC.BAT file the command `mode mono` to activate your monochrome monitor; use the command `mode co80` to activate your color monitor.

If you have a multifunction graphics board and you're not sure whether it's configured for monochrome or color, include the `mode mono` command in your AUTOEXEC.BAT file. This command will prevent your monochrome monitor from burning out if your board is set for color.

If you use a Hercules Graphics Card, or something comparable, you should probably include the command `hgc diag` in your AUTOEXEC.BAT file to initialize the board.

Warnings

Watch out for application programs that include an AUTOEXEC.BAT file on their distribution diskettes. Some programs cannot be executed properly without a series of commands; these commands are contained in the AUTOEXEC.BAT file supplied with the program. If you install these programs in subdirectories on a hard disk, the AUTOEXEC.BAT file supplied with the application will not be executed. One remedy is to rename the application's AUTOEXEC file, move it to your \BATCH directory, and add a CD command to the beginning to make the application's directory current. Another remedy is to make the program's directory current and then issue the command `autoexec`.

Be even more wary of applications that automatically rewrite your hard-disk AUTOEXEC.BAT file when you install them. Some applications make changes that can be annoying, such as instructing DOS to load the application before you have initialized your screen. Other applications might delete your existing AUTOEXEC file altogether, thereby wiping out your initialization routines. Programs that change your AUTOEXEC.BAT file usually tell you so. If so, be sure to examine the file before you reboot your computer, and change it back if necessary.

Tips

In DOS 4.0 you can install TSRs in memory by using the Install= command in your CONFIG.SYS file, instead of using your AUTOEXEC.BAT file for the purpose. This has the advantage of installing the TSRs first in memory, which may make more efficient use of memory. However, it also means that you can't unload these TSRs without rebooting the computer. See CONFIG.SYS File, Install=, TSRs.

For further information, see Batch Files, Call, Command, Directories, Echo, ErrorLevel, Exist, For . . . In . . . Do, GoTo, If, Path, Pause, Rem, and Shift. Consult the manuals that came with your hardware to determine whether your hardware requires special commands in your AUTOEXEC.BAT file.

AUX. *See* Device Names

AUX is an alternative name for the first serial port.

Backup

Syntax

backup *source*[*filename.ext*] *target* [*/switches*]

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3¦
IBM DOS 4.0

Overview

The Backup command is used primarily for backing up hard disks to diskettes. In DOS 3.0 and later you can also use Backup to perform these types of backups:

- Diskettes to other diskettes
- Diskettes to a hard disk
- Hard disks to other hard disks
- Hard disks to themselves
- Hard disks or diskettes to a network file server

Types of Backups The Backup command has many limitations, and there are better ways to accomplish most of these tasks. (See Hard Disks, Backing Up and Restoring.)

When you use the Backup command, you have a range of choices regarding what to back up. You can back up

- All of a hard disk
- A particular branch of a subdirectory tree
- Files changed since a given date (and time)
- Particular files
- Files that have changed since you last did a backup

These choices represent three types of backups: full, incremental, and partial.

A full backup is a backup of every file on the hard disk. It is like a snapshot of your complete file structure at the time you did the backup.

An incremental backup is a backup of files that were created or changed since the previous backup. Such files will have the archive attribute set (see File Attributes). When you execute the Backup command, it clears the archive attribute of each file after it is backed up. This allows you to make incremental backups without including files that have not changed. You should do an incremental backup about every three days; back up daily if you use your computer intensively.

A partial, or selective, backup is a backup of a particular set of files or a particular branch of your directory tree. For example, you might want to back up all files relating to a given project when you complete it. Or, you might want to back up the files from different projects onto separate sets of diskettes. You may find it easier and faster, however, to make this type of backup using the Copy, Replace, or Xcopy command.

For the following discussion, we'll assume that the hard disk is drive C and that the backed up files will be written on a diskette in drive A. Other combinations are possible, however. If you want use other media, substitute both the appropriate drive name for A and the appropriate backup medium for diskettes.

If your version of DOS can back up to a hard disk, and you choose to do so, the Backup command will create a directory called \BACKUP one level below the root. It will also create a tree structure subsidiary to that directory that matches the source. All the backup files will appear there, in the appropriate subdirectories.

Some General Characteristics of the Backup Command The Backup command expects the target media to be formatted and will abort if it encounters an unformatted diskette. Therefore, you must have enough formatted diskettes to complete the task before you begin. The only exceptions are DOS 3.3, which allows you to format as you go, and

DOS 4.0, which will format the backup diskettes automatically if needed. See the Tips section for details.

You cannot change drives in midstream. Once you begin a backup, all the backup files will be written to the drive you specified in your command.

The Backup command changes the structure of the files on the target diskette. The file names on the target will be the same as those on the source, but the files themselves include the names of the source drive and directory along with the source file. Thus, you cannot use backup files unless you first return them to their original condition with the Restore command.

There is another reason you cannot use backup files without restoring them first. Backup writes to a diskette until it is full, writing only part of a file at the end if necessary. When you insert the next diskette, Backup writes the remainder of the partly written file to that diskette. Thus, some files will be split across backup diskettes. This feature allows you to back up very large files.

Figure 5 shows the directory of a backup diskette made from a single directory of a hard disk. For comparison, Figure 6 shows the source directory on the hard disk. Note the file BACKUPID.@@@. Every backup disk includes a file by this name, which identifies the disk as a backup disk and notes its place in the backup series. Note also that every file on the backup diskette is longer than the original as listed in the hard disk subdirectory. The difference comprises the control information that the Restore command will use to recreate the files in their original location.

```
 Volume in drive B is BKUP DISK 1
 Directory of  B:\

BACKUPID @@@       128   9-25-87   4:02p
MEMSPEED COM      1792   5-02-86  12:03a
FRP      EXE     79327   8-17-86   9:31a
FAST     DOG       317   2-13-87   9:06a
DOG      EXE     36704   6-06-86   4:17p
BACKSTAT EXE     13186   8-18-86  10:26p
RAMTEST  COM      1202   7-12-87  10:42p
SPEED    COM     26267   6-01-86  11:55a
CLUSTER  COM      1792   6-15-87   2:00a
RAMTEST  ECT      1383   6-18-87  12:43p
RAMTEST  OVL     27596   9-17-87  12:38p
FULL     DOG      2653   9-18-87   5:18p
        12 File(s)     164864 bytes free
```

Figure 5: Directory of a backup diskette

```
 Volume in drive C is JK'STURBOAT
 Directory of  C:\UTIL\PFMNCE

MEMSPEED COM      1664   5-02-86  12:03a
FRP      EXE     79199   8-17-86   9:31a
FAST     DOG       189   2-13-87   9:06a
DOG      EXE     36576   6-06-86   4:17p
BACKSTAT EXE     13058   8-18-86  10:26p
RAMTEST  COM      1074   7-12-87  10:42p
SPEED    COM     26139   6-01-86  11:55a
CLUSTER  COM      1664   6-15-87   2:00a
RAMTEST  ECT      1255   6-18-87  12:43p
RAMTEST  OVL     27468   9-17-87  12:38p
FULL     DOG      2525   9-18-87   5:18p
        13 File(s)   9125888 bytes free
```

Figure 6: Source directory of a backup diskette

Release 3.3 and 4.0 use a different format for backups. In these DOS releases, each backup diskette contains only two files—BACKUP.*nnn* and CONTROL.*nnn*. The first file contains all the backed up files on the disk, and the second contains the control information stored in the backup files in previous releases. This makes the backups more compact, thereby minimizing the unused space on the target diskette.

Procedure/ Examples

Backup is an external command; using it requires having the DOS program BACKUP.COM (or BACKUP.EXE) on a disk in your computer. If you use DOS 2, it must be either on the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:backup c:\ a:
```

If you use DOS 3.0 or later and the program is not on the current directory or the path, preface the command with a drive name and/or path name; for example,

```
c:\dos\backup c:\ a:
```

You must always specify a source and a target for the Backup command. The target is usually a drive name; the source may be a drive, a directory, a file name, or any combination of them. If you specify a drive but not a directory, the backup will begin with the current directory of that drive. If you specify a directory but not a drive, the backup will begin with the current directory of the current drive. If you specify a file name, only files matching the name you specify will be backed up. You can use wildcard characters in file names. (See Wildcard Characters.)

The Full Backup The first step in any process using the Backup command is to ensure that you have enough formatted target diskettes. To guarantee maximum capacity, use the type of diskette for which the drive is designed. (In other words, don't use any switches to the Format command except /v.)

Of the three types of backup, it's easiest to estimate the diskettes needed for a full backup. Read a directory on your hard disk and note the number of bytes free in the last line of the directory display. Divide this number by 1024 to see how many kilobytes are free. Next, determine the total size of your hard disk in kilobytes. (Since you probably know it in megabytes, you can just divide the nominal capacity by 1000.) Now subtract the number of free kilobytes from the capacity of your hard disk. This tells you how many kilobytes of data you want to back up. Finally, divide the result by the nominal capacity of the backup diskettes you will use. Format that number of diskettes, then format a few extra since nominal capacity is greater than the true capacity. (See Diskettes, Structure of.)

Now you are ready to begin. To back up everything on the hard disk, log onto it and enter the command `backup c:\ a:/s`. The /s (subdirectories) switch tells the command to back up all subdirectories of the specified source directory. You will see this prompt and message:

```
Insert backup diskette 01 in drive A:
Warning! Files in the target drive
A:\ root directory will be erased
Strike any key when ready
```

This message varies somewhat in different releases of DOS, but when you press a key, Backup begins copying files from the hard disk, displays each file's name as it is copied, and clears the archive attribute when the copy is complete. When the disk is full, you will be prompted to insert the next diskette.

As each diskette becomes full, label it as a backup diskette. Number the diskettes sequentially so that you can find the diskette you need when you want to restore a file.

If this was your first full backup, repeat the procedure. The Backup command is notoriously unreliable, and, to make matters worse, disks

sometimes go bad. If you need to restore a file from a damaged diskette, you will be glad you made the second copy.

Incremental Backups To perform an incremental backup, you use the /m (modified) switch as well as the /s switch, resulting in the command

```
backup c:\ a:/s/m
```

This instructs the Backup command to back up the entire hard disk, ignoring all files that have not changed since the previous backup.

Do not use the diskettes from one of your full backup series; you should keep both sets in case you need to restore a file. Instead, begin a third series for your incremental backups (incremental backups rarely require very many diskettes). Label your diskettes so you can tell them apart from your full backup series, and number them in order.

After the first incremental backup, you may prefer to keep your incremental backups in a single series rather than starting a new series each day. This can save you quite a few diskettes, since each day's series will probably leave a diskette only partly full. To continue the same series, use the /a (append) switch for each subsequent incremental backup:

```
backup c:\ a:/s/m/a
```

This command tells Backup to make copies of all modified files, in all subdirectories, and add them to the diskette in drive A.

When you use the /a switch, the message

```
Insert backup diskette in drive A:
Strike any key when ready
```

appears. Notice that the warning about erasing the root directory does not appear. This allows you to add files to the last diskette of your previous backup series. The message will appear again after you have filled that diskette.

Partial Backups You need not back up an entire disk at once. You can specify a directory or even a file to back up:

```
backup c:[path\][filename.ext] a:
```

You can even specify a group of files by including wildcard characters in the file name. However, you can back up several groups of files by name

by simply issuing several backup commands, each with the appropriate file name. Suppose, for example, you are working on a budget plan that includes both word-processor and spreadsheet files with extensions .DOC and .WRK, respectively. If you keep these files in a common directory called \BUDPLAN, you can back up these two sets of files with the following commands:

```
backup c:\budplan\*.doc a:
backup c:\budplan\*.wrk a: /a
```

The first command will back up only the .DOC files. The second command adds the .WRK files to the same backup diskette.

Selecting Files by Date and Time Backup also gives you the option of making partial or incremental backups by date by using the /d (date) switch. Here is how you might use this option: Suppose you keep your files in the same directory as your software (a practice you should probably avoid). Your data files probably have later dates than your software. Thus, using a Backup command with a date later than the date on your software files will back up only your data files. (Of course, if you change your software configuration, the software will probably rewrite one of its files. The updated program file would also be backed up among the data files.)

To use the date switch, follow the switch with a colon and the date in your country's date format. For the United States, this format is *mm-dd-yy* or *mm/dd/yy*; for example,

```
backup c:\budplan a: /d:03/17/89
```

With MS-DOS 3.2 and DOS 3.3, you can also elect to back up only files created after a given time. To use the time switch, follow the switch with a colon and the time in your country's time format. This format is the one used in the directory. Thus, in the United States, if your target time is 9 in the morning, enter `9:00a`. Since the Backup command treats time and date independently, you will probably want to use the /t switch and the /d switch together. Otherwise, Backup will copy all files with the directory time you specify or all files with a later time, regardless of the date they last changed. Your backup command might look something like this:

```
backup c: a:/s/d:6-20-89 /t:1:00p
```

The date and time switches are independent of the /m switch. If you want to back up only *modified* files that last changed after a given date, you need a command such as

```
backup c: a:/s/m/d:6-20-89 /t:1:00p
```

Use a command of this type to distinguish between software and data in a directory for a partial backup.

Warnings

Unless you have DOS 3.3 or later, it is vital that you have enough formatted diskettes; otherwise, you may have to repeat the entire process.

The Backup command is very slow, especially if you have turned Verify on. However, it is especially important to turn Verify on when you do a full backup. The Backup command is not especially reliable, so if you do not turn Verify on, you risk not being able to restore a file when you need to. Even Verify provides only limited error-checking. Therefore, you may find it expedient to invest in commercial backup software (or, for large hard disks, a tape backup system). Most of these systems are faster and more reliable than the Backup command.

Do not issue a Backup command when you have a Join, Subst, or Assign in effect: you may be unable to restore files to their proper directories if you do so.

Tips

When to Back Up Your Hard Disk You should do some kind of backup every day. At the very least, back up each day's modified files at the beginning or end of your work day.

Formatting As You Go With DOS 3.3, you can have the Backup command format disks as you proceed. To do so, use the /f switch; for example,

```
backup c:\ a:/s/f
```

For this switch to work, FORMAT.COM must be either in the current directory or on the search path (see Path). If it is not, Backup will abort when it encounters an unformatted disk.

With DOS 4.0, you don't even need to add a switch to the Backup command to format as you go. This version of Backup will automatically format unformatted backup disks. However, it still requires FORMAT.COM to be in the current directory or on the search path. In addition, you must be using the type of diskettes for which the drive was designed. If you

try to use 720K diskettes in a 1.44-megabyte drive, or 360K diskettes in a 1.2-megabyte drive, FORMAT will attempt to format the diskette to the capacity of the drive, resulting in numerous bad sectors and an unreliable backup diskette. Backup will not attempt to format a hard disk if a hard disk is the target drive for the backup.

When it's time to restore a file, you'll find it useful to have a record of which files appear on which backup diskette. If you use a version of MS-DOS earlier than 3.2, or a version of PC or IBM DOS earlier than 3.3, you may want to direct the output of the Backup command to the printer:

```
backup c:\ a:/s>prn
```

(See Redirection.) This will give you a printed record of all the files as they are backed up.

All messages regarding when to change disks will also appear on the printer instead of the screen. However, since messages of the form

```
*** Backing up files to drive A: ***
Diskette Number: 12
```

will also be printed, you will know exactly which disk contains which files.

To have your messages appear both on the screen and on paper, press Ctrl-P or Ctrl-PrtSc before you start, instead of redirecting the output. *Do not*, however, attempt to redirect the output to a file. If you ignore this advice, prompts to change disks, error messages, and other information you should know about will not appear on the screen. You will not know when to change disks unless you listen very carefully to your diskette drives.

Keeping a Record of Backup Files The DOS 3.3 and MS-DOS 3.2 versions of Backup can create a log file for you so that you don't have to redirect the output to the printer. Adding the /l switch to your Backup command creates a file called BACKUP.LOG in the root directory of the disk you are backing up. You can give the file a different name (which might be useful if you want separate files for your full and incremental backups) by including a file name in your command line:

```
backup c:\ a:/s/l:fullback.log
```

```
[9-25-1988 16:02:10p]
      1  /UTIL/PFMNCE/MEMSPEED.COM
      1  /UTIL/PFMNCE/FRP.EXE
      1  /UTIL/PFMNCE/FAST.DOG
      1  /UTIL/PFMNCE/DOG.EXE
      1  /UTIL/PFMNCE/BACKSTAT.EXE
      1  /UTIL/PFMNCE/RAMTEST.COM
      1  /UTIL/PFMNCE/SPEED.COM
      1  /UTIL/PFMNCE/CLUSTER.COM
      1  /UTIL/PFMNCE/RAMTEST.ECT
      1  /UTIL/PFMNCE/RAMTEST.OVL
      1  /UTIL/PFMNCE/FULL.DOG

[9-25-1988 17:26:00p]
      1  /UTIL/TOOLS/BULKERAS.DOC
      1  /UTIL/TOOLS/BULKERAS.EXE
      1  /UTIL/TOOLS/COPYIIPC.EXE
      1  /UTIL/TOOLS/NOGUARD.COM
      1  /UTIL/TOOLS/CRLF.EXE
      1  /UTIL/TOOLS/ARC.EXE
      1  /UTIL/TOOLS/SOFTEN.COM
      1  /UTIL/TOOLS/WSFILTER.EXE
      1  /UTIL/TOOLS/ASM.EXE
      2  /UTIL/TOOLS/EXE2BIN.EXE
      2  /UTIL/TOOLS/LINK.EXE
      2  /UTIL/TOOLS/LINK86.EXE
      2  /UTIL/TOOLS/MASM.EXE
      2  /UTIL/TOOLS/DISCATT.EXE
      2  /UTIL/TOOLS/MYLIB.EQU
      2  /UTIL/TOOLS/MYLIB.MAC
      2  /UTIL/TOOLS/RMAC.EXE
      2  /UTIL/TOOLS/STRING.COM
      2  /UTIL/TOOLS/FILEINFO.FI
      2  /UTIL/TOOLS/BASIC.KEY
      2  /UTIL/TOOLS/QR.KEY
      2  /UTIL/TOOLS/QUICKREF.ASC
      2  /UTIL/TOOLS/QUICKREF.BAS
      2  /UTIL/TOOLS/WORDSTAR.KEY
      2  /UTIL/TOOLS/WORDSTAR.PRN
      2  /UTIL/TOOLS/MENUMAKR.EXE
```

Figure 7: A sample BACKUP.LOG file

Figure 7 shows a sample backup log. It includes

- The starting date and time of the backup
- The complete path name of each file that was backed up (separated by forward slashes instead of the more usual backslashes).

At the beginning of each line is a numeral. This is the number of the disk in the backup series that the backed-up file will appear on.

The first part of Figure 7 is the log file for the backup illustrated in Figures 5 and 6. Notice that when you use the /a switch, Backup adds the record of each additional backup series to the existing log.

Incompletely Filled Backup Diskettes Although your diskette may have adequate space, Backup sometimes asks you to change diskettes. This most often happens when the backup diskette contains many short files. Each file requires a new entry in the directory, which has limited space. Thus, if the directory becomes full, no additional files can be written, even though there may still be room on the diskette.

Backup copies files only to the root directory of the backup diskette. Although you cannot increase the number of files that backup will copy, you can create a backup system using the Xcopy command, which creates subdirectories as it proceeds (see Xcopy and Hard Disks, Backing Up and Restoring).

Backup has exit codes that can be tested with the ErrorLevel command (see ErrorLevel).

For further information, see Restore and File Attributes. For alternative ways of backing up a hard disk and a discussion of backup strategies, see Hard Disks, Backing Up and Restoring.

Batch Files

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

A batch file is a script for DOS that contains a series of commands for DOS to perform. A batch file can contain any command that DOS can execute. You can use batch files to:

- Execute any sequence of commands that you use often
- Create alternative names for commands
- Use the same sequence of commands on different files
- Create convenience commands
- Simplify hard-disk navigation
- Save and restore default conditions

BATCH FILES B

Batch files are powerful tools for streamlining your computer work.

Batch files can contain any of the internal DOS commands as well as commands that execute programs. There are also several special commands for use exclusively in batch files. These special commands make up a relatively simple programming language, which, among other properties, allows you to include variables to which you can give a value when you execute the file.

Batch files are executed one line at a time. DOS reads a command line from the batch file, executes it, and then returns to the batch file for the next line. Unless you interrupt it with Ctrl-Break, DOS continues until it has executed every command in the file. As the name implies, batch files do not permit interactive input. The only choice you have is whether to continue.

AUTOEXEC.BAT is a special batch file. AUTOEXEC.BAT is executed automatically when you start up your computer if it is on the root directory of the boot disk. Use this file to take care of all your housekeeping before starting your application (see AUTOEXEC.BAT File).

Procedure

Creating and Invoking Batch Files A batch file is an ASCII text file containing a sequence of DOS commands, special batch programming instructions, and executable file names. You can create batch files with the Copy CON command, with EDLIN, or with a word processor in ASCII or nondocument mode (see Copy CON and EDLIN). When you save the file, you must give it the extension .BAT.

You should not give a batch file the same name as a DOS internal command; neither should you give it the same name as a .COM or .EXE file on the same directory. DOS will not execute the batch file if you do so.

As with other DOS commands, you invoke a batch file by typing its name on the command line. You needn't type the extension. If you include replaceable parameters (discussed under Using Batch Programming Instructions on the next page) in your batch file, parameters you enter on the command line will also be processed by the file.

The same rules that govern commands typed on the command line govern commands in batch files. Thus, if a command is a program name, the program file must be in the current directory or on

the search path. If it is not on the current drive, you must precede it with a drive name; if it is not in the current directory or on the search path (and you use DOS 3.0 or later), you must precede it with a path name. If you use DOS 2, you must include a CD command in your batch file before any command that is not in the current directory or on the search path.

Executing a Batch File from Another Batch File To have a batch file execute another batch file and then continue, preface the name of the second batch file with either the Call command or the Command command (see Call and Command). If you simply invoke another batch file, execution will branch to the second batch file and not return to the first.

If the command invoking the second batch file is the last command in the first one, an extra `command /c` or Call command is unnecessary. Execution will proceed to the second batch file automatically.

Using Batch Programming Instructions The following commands make up the batch programming language:

Call
Echo
For . . . In . . . Do . . .
GoTo *label*
If
If ErrorLevel
If Exist
If Not
Pause
Rem
Shift

See the respective entries for full explanations of these commands. These commands allow you to

- Create loops
- Branch from one part of a batch file to another, or even to a different batch file
- Interrupt the program before it ends
- Display text while a batch file is executing

- Pause for confirmation by a user
- Test for various conditions, and base the subsequent action on the result

Passing Parameters There is no special command for the function, but you can pass parameters to a batch file by entering them on the command line.

To pass parameters to a batch file, you must put *replaceable parameters* in your batch file. A replaceable parameter is a symbol that takes on the value of an argument typed on the command line. It is represented by a percent sign followed by a number from 0 to 9.

You can use up to ten different replaceable parameters in a batch file and pass arguments to nine of them from the command line. The first parameter on the command line replaces %1, the second replaces %2, and so on. The replaceable parameters can represent drive names, path names, file names, switches, or any other acceptable command-line parameters; they can even represent parts of command-line parameters.

The parameter %0 is a special parameter representing the name of the batch file that contains it. You can take advantage of this parameter to perform some interesting tricks. First, if the batch file is on the search path or in the current directory and there is no .COM or .EXE program of the same name in the current directory (or earlier on the search path), the batch file will invoke itself and run again. You can thus use the %0 parameter to create a loop.

You can also use it to invoke a .COM or .EXE file of the same name by controlling which directory is current within the batch file. Thus, for example, if you used Lotus 1-2-3, which cannot be invoked properly from anywhere except its home directory even if it is on the search path, a batch file called 123.BAT, of the form

```
CD\LOTUS
%0
```

would successfully invoke it. I leave more complex uses to your imagination.

Examples

Executing a Fixed Sequence of Commands Repeatedly You can use a batch file to execute a sequence of commands by typing only a

single command. You can also ensure that a command is always executed with a particular set of parameters.

Suppose you regularly make backup copies of data diskettes, and that you have a hard disk and two diskette drives of the same type. You want the files on at least one copy to be written contiguously, and you also want to be sure that your copy is accurate. To ensure that your files are contiguous, you want to use Xcopy or Copy, rather than DiskCopy; however, unlike the other commands, DiskCopy formats diskettes as needed. The batch file DATABAK.BAT, shown below, will copy all your files in unfragmented form. All you need do is enter the command

```
databak
```

DATABAK.BAT

COMMANDS	COMMENTS
`c:`	Makes C the current drive
`cd\dos`	Makes \DOS the current directory
`format b:`	Formats the target diskette
`chkdsk b:`	Ensures the integrity of the target diskette
`copy a:*.* b: /v`	Makes and verifies the copy
`dir b:`	Displays the result

In addition to repeating a sequence of commands, a batch file can ensure that you include all the switches and parameters you need.

Although it's a good policy to run ChkDsk regularly to guarantee that you don't have any lost clusters or cross-linked files, it's easy to forget to include the /f switch, which authorizes ChkDsk to make the repairs. You can create a file called FIX.BAT which will ensure that you remember the switch. This file would contain only the line

```
chkdsk /f
```

To run FIX.BAT on different drives, you might create separate versions of the file called FIXA.BAT, FIXB.BAT, and so on. Instead, however, you can use a replaceable parameter, as described above. Thus, to use FIX.BAT on any drive, create the file so that it reads

```
chkdsk %1: /f
```

To lock out bad sectors on drive A, for example, enter the command

```
fix a
```

You needn't enter the colon because it's included in the file. If you do enter the colon, you will see the message

```
A:\: File not found
```

at some point because DOS will look for a file called : in the root directory of drive A.

Creating Alternative Command Names You can use batch files to shorten command names or to give yourself the option of using different names. For example, you might want to shorten DiskCopy and DiskComp. You could create a file called DCPY.BAT that included the single line

```
diskcopy a: a:
```

for a system with a hard disk and a single diskette drive, or

```
diskcopy a: b:
```

for a two-diskette drive system. Similarly, you could shorten DiskComp to Dcmp by creating a file called DCMP.BAT that simply said

```
diskcomp a: a:
```

or

```
diskcomp a: b:
```

Repeating Commands on Different Files Suppose you use WordPerfect, but regularly share files with someone who uses WordStar. If you don't use extensions in the names of your word-processing files, you can use WordPerfect's Convert utility to create files that WordStar can read from your originals. However, since you make the same conversion regularly, it would save time to do it automatically, bypassing the menus. The Convert utility allows you to do this if you provide all the information it needs on the command line; however, it's not easy to remember the menu options by number, and that's where a batch file comes in handy. A one-line batch file called WP2WS.BAT can remember the menu options for you:

```
convert %1 %1.ws 1 3
```

Again, a replaceable parameter is used; this time it stands for the name of the file to convert. To accomplish the conversion, type

```
wp2ws filename
```

entering the name of the file you want to convert. The single line invokes the utility, tells it to convert the file to a new file called *FILENAME*.WS, and selects option 1 (WordPerfect) from the Input file menu and option 3 (WordStar) from the Output file menu.

If you give extensions to your file names, this program will generate an error message and you will have to use the menus. This is because the complete file name—for example, REPORT.WP—will replace the second %1. The batch file includes a .WS extension for the output file name. Thus, the text `%1.ws` turns into REPORT.WP.WS, which is not a legal DOS file name.

However, you can still manage your format conversions with a batch file by including separate replaceable parameters for the file name and its extension:

```
convert %1.%2 %1.ws 1 3
```

The parameters %1 and %2 are separated by the period that always separates a file name from its extension. With this syntax, the file name will replace %1 and the extension will replace %2. See For . . . In . . . Do and Shift to learn ways of executing a command such as `Wp2ws` on a group of files.

Creating Convenience Commands Many procedures are unnecessarily long. For example, if you want to switch the names of two files, you can't rename one to the other because both already exist. Consequently, you must use a third name temporarily and enter three commands:

```
ren file1.ext temp
ren file2.ext file1.ext
ren temp file2.ext
```

You could simplify the process by placing these commands in a batch file called SWAPNAME.BAT and typing the command

```
swapname file1.ext file2.ext
```

Of course, unless you always want to switch the same two names, you should use replaceable parameters:

```
ren %1 temp
ren %2 %1
ren temp %2
```

Simplifying Hard-Disk Navigation There are two ways that batch files can help you get around your hard disk: first, you can use batch files

to execute programs on directories that are not current, and second, you can use batch files to create aliases for long path names.

To use batch files for navigation, keep them all in their own directory—perhaps one called \BATCH. Put this directory on the search path near the beginning (see Path). If you take these steps, you can regard the batch files in the \BATCH directory as an extension of your search path.

Plan your search path at the same time as you set up batch files for executing programs. Presumably, you will place the directions for the applications you use most often on the path. If they use ancillary files, you may also need to append their directories (see Append). However, you probably have some programs that you use less often. Such programs are more efficiently called by using a batch file. Let's consider several circumstances and look at batch files appropriate for each.

Assume you use DOS 2. The only way to execute programs on noncurrent directories in DOS 2 is to place them on the search path. However, you still won't be able to execute some programs. Without path-extension utilities (forerunners of the Append command), some programs can't find their ancillary files. Even so, you can use batch files to simplify execution.

For programs not on the search path, use batch files of the general form

```
echo off
cd\progdir
prog %1 %2 %3
cd\
```

In this file, *progdir* is the name of the directory where your software resides, and *prog* is the name of the program. This simple file makes the *progdir* directory current, executes *prog*, then returns you to the root directory. If you normally invoke your program with switches, file names, or other parameters, you can enter these parameters on the command line after the name of the batch file. They will be passed to the replaceable parameters in the file.

You can make these files considerably more elaborate. For example, you can have them copy the new file to a backup disk, clear the screen, and show the directory (or whatever else is appropriate).

DOS 3.0 and later gives you considerably more freedom in executing files since you can execute programs with a command of the form

drive\path\progname

This means that you can invoke a program—provided it either doesn't need supplementary files or knows how to find them—from anywhere in your system. However, even a command of this variety is more difficult to enter than a simple two-letter batch file name.

The ability of DOS 3.0 and later to execute files on noncurrent directories increases flexibility in constructing batch files. For example, if you use either a keyboard macro program or a mouse, you can have the batch file load the appropriate files. BUSLTR.BAT, the program that follows, illustrates this technique by loading WordStar 4.0 with a set of defaults for business letters. (Assume that WordStar's directory, \WORDSTAR, is on the search path and that the business letters are in a directory called \LETTERS\BUSINESS.) To invoke BUSLTR.BAT, type

```
busltr filename
```

where *filename* is the name of the file you want to open or create. The batch file automatically adds the extension .LTR.

BUSLTR.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`key e:ws /ml`	Loads the SuperKey macro file for WordStar
`cd\letters\business`	Makes the \LETTERS\BUSINESS directory current
`ws %1.ltr`	Loads WordStar with the specified file open
`key /mc`	Clears the SuperKey macro file
`del %1.bak`	Deletes the backup copy of the letter
`dir /p`	Shows the directory
`cd\`	Makes the root directory current

Let's address another type of problem. Suppose you have a complex directory tree with deeply nested subdirectories (use the directory tree shown in Figure 28 in the Directories entry as an example). In this case, you want to be able to invoke a program and have it open a file on one of your nested directories; however, you don't want to type a command such as

```
fw \invoices\south\qtr3\august
```

to invoke, say, Framework III, and have it open a file on the directory \INVOICES\SOUTH\QTR3. Instead, you could include all the pertinent material in a batch file called FWIS38.BAT:

```
fw \invoices\south\qtr3\august
```

You can make this batch file more powerful if you use the directories often by using a file containing only this line:

```
%1 \invoices\south\qtr3\%2
```

Call this file IS3.BAT. With a file such as this, you can invoke any program on the search path with a command such as

```
is3 fw august
```

You can also use batch files to help with nested software directories. A one-line batch file can invoke a program on the lowest level directory. For example, suppose you have two printers, and you have the utilities for your Epson printer in a directory called \UTIL\PRINTER\EPSON. You can invoke those utilities by using a batch file called EPS.BAT, containing the line

```
c:\util\printer\epson\%1
```

You would then type a command such as

```
eps italic
```

Finally, if you just want to make deeply nested directories current, your batch file could contain a line such as

```
cd\invoices\south\qtr3
```

Let's call this file GOIS3.BAT. To repeat an earlier example, you can now open the AUGUST.FW3 file in Framework III by typing these two commands:

```
gois3
fw august
```

An additional advantage is that \INVOICES\SOUTH\QTR3 will be the default directory when Framework III comes up.

Saving and Restoring Defaults You can use batch files to save default conditions so that you can restore them later. Suppose your path is so long that you cannot add any new directories to it. To use some software, however, you need to put a directory on the path that isn't normally there, but you want your old path back when you're done with

that application. One way to accomplish this is to keep your regular path in a batch file called REGPATH.BAT. When you invoke the application that requires the special path, include a Path statement in it. For example, you might want to invoke a style checker by using a batch file such as STYLE.BAT, which follows.

STYLE.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`path c:\dos;c:\batch;c:\style`	Sets temporary path
`style %1`	Invokes application, opens file
`regpath`	Restores regular path

Notice that the temporary path includes the \DOS and \BATCH directories as well as the one you need to use your application. You must have the \BATCH directory on the path to invoke REGPATH.BAT, which is presumably in that directory (see Directories). Also, it's always a good idea to keep DOS available.

As the Path entry shows, you can use the same technique to store prompts and append statements. For an alternative method, see Set.

Switching among Configurations Sometimes you need to run an application that uses a different configuration from the one you normally use. Let's say you have a program such as AutoCAD, which requires at least 512K bytes of memory. However, when you're not using AutoCAD, you use a group of device drivers such as DRIVER.SYS, several copies of VDISK.SYS, and several TSR utilities. These files all take up some memory that AutoCAD could use, so you need a way of getting rid of them temporarily. On the other hand, you don't want to rewrite your AUTOEXEC.BAT and CONFIG.SYS files every time you use AutoCAD.

The solution is to have two sets of AUTOEXEC.BAT and CONFIG.SYS files. When you're not using AutoCAD, the versions you need to run AutoCAD are called AUTOEXEC.AC and CONFIG.AC, respectively. You switch configurations by using a pair of batch files, USEAUTO.BAT and NOAUTO, which follow. USEAUTO.BAT simply renames the AUTOEXEC and CONFIG files so that the set for AutoCAD has the proper names.

Similarly, NOAUTO renames these files so that they will work with everything else. The Ctrl-G in the last line sounds a beep to alert the user to reboot the computer. (Unfortunately, there is no way other than rebooting to get DOS to read the CONFIG.SYS file.)

USEAUTO.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`ren config.sys config`	Renames CONFIG.SYS so it won't be used
`ren config.ac config.sys`	Renames CONFIG.AC so it will be used
`ren autoexec.bat autoexec`	Renames AUTOEXEC.BAT so it won't be used
`ren autoexec.ac autoexec.bat`	Renames AUTOEXEC.AC so it will be used
`echo GReboot computer to load AutoCAD configuration`	Tells the user what to do

NOAUTO.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`ren config.sys config.ac`	These lines restore the original names
`ren config config.sys`	
`ren autoexec.bat autoexec.ac`	
`ren autoexec autoexec.bat`	
`echo ^GReboot computer to restore normal configuration`	Tells the user what to do

Warnings Every line in a batch file must begin at the left margin of the screen. If you precede a command with spaces or tabs, the command will not execute,

and it will appear on the screen even if Echo is off. The effect on subsequent lines is unpredictable.

You cannot include text in a batch file that is not part of a command. The batch language includes the Rem command for adding notes to a batch file (see Rem).

In DOS 3.0, you cannot remove a disk from which you are executing a batch file. Even if the same batch file is on the disk you replace it with, you will see the message

```
Batch file missing
```

Either rewrite your batch file so that you don't have to switch disks, or upgrade to a later version of DOS.

Tips

Batch files differ from other executable files (.COM or .EXE files) in one important respect—they are not loaded into memory. When you invoke a batch file, DOS reads a line of the file, executes it, and then returns to the file to read the next line. Thus, the batch file must remain available to DOS.

If you switch disks before the batch file finishes, you will remove the batch file from the drive from which you invoked it. DOS won't know where to look for the next command, and will display the prompt

```
Insert disk with batch file in drive A:
Then press any key when ready
```

You won't encounter this message if you also place a copy of the batch file on the second disk. If the disk is in the same drive from which you invoked the batch file, DOS won't know the difference, and execution will continue without interruption.

You can interrupt a batch file while it's executing by pressing the Break key (Ctrl-C or Ctrl-Break). As soon as DOS next checks for a Break (see Break), it displays the message

```
Terminate batch job (Y/N)?_
```

If you press Y and Return, execution will end. If you press N and Return, it will continue. If you press any other key and press Return, the message will appear again. If you do not press Return, nothing will happen.

If you're not sure which directory will be current when you execute a batch file, your batch file can make the proper directory current. Alternatively, if you use DOS 3.0 or later, you can include a complete drive and path name as part of any batch file command.

Because DOS must read from the batch file after it executes each line, batch files execute relatively slowly and are especially slow on diskettes. If you use a RAM disk, you can speed up batch files by copying them to the RAM disk and executing them from there (see VDISK.SYS, RAMDRIVE.SYS).

You can extend as far as you wish the principle of letting replaceable parameters replace parts of command-line arguments. For example, your batch file might include the following string:

```
%1:\%2\%3.%4
```

To this strange annotation you can pass a drive name, a path name, a file name, and an extension, each as a unit. For this to work properly, the arguments on the command line must be something like this:

```
b letters\business * doc
```

If you treated the entire series as a unit, such as

```
b:\letters\business\*.doc
```

the unit would replace only the %1 parameter, resulting in the argument

```
b:\letters\business\*.doc:\\.
```

DOS finds this unintelligible, and you probably do too.

As an alternative to using batch files for shortening path names, use the Subst command (DOS 3.1 and later). This command substitutes a drive name for a path name (see Subst).

Including blank lines in a batch file does not cause any problems; in fact, if some commands do not execute accurately, you may be able to cure the problem by placing a blank line after the problem command. You are most likely to encounter this problem on computers with high clock speeds.

For further information, see AUTOEXEC.BAT File, Call, Echo, Error-Level, Exist, For . . . In . . . Do, GoTo, If, Path, Pause, Rem, and Shift.

Boot Record

Overview

The boot record is the first sector (side 0, track 0, sector 1) of any disk or hard disk partition formatted by DOS. The boot record contains a small program that starts your computer and tells the BIOS the names of the hidden system files that must be loaded before anything else. In addition, it contains a great deal of information about the disk, including

- The version of DOS with which the disk was formatted
- How many bytes are in each disk sector

- How many sectors are in each cluster, track, and disk
- How many sectors are reserved for two copies of the file allocation table and the root directory
- How many copies of the FAT are on the disk (normally two) and how many sectors it occupies
- How many entries the root directory can hold
- How many sectors are on the disk
- How many formatted sides (or cylinders) the disk has
- Messages to display if the disk is not a system disk and you try to boot your system from it

Warnings

If the boot record of a disk is corrupted, DOS cannot read the disk, even if the data are intact. If you are sure you know how the disk is formatted, you may be able to salvage the disk. Use DEBUG to load into memory the boot record of another disk formatted identically. Then use DEBUG's Write command to write that record in the first sector of the damaged disk. This procedure should make the disk readable again, unless the first sector is physically damaged. If this technique fails, you may be able to recover the data to write it to another disk with a commercially available file-recovery utility.

Tips

If you try to use the Sys command (see Sys) on an empty disk and you see the message

```
No room for system on destination disk
```

the disk was probably formatted by a version of DOS other than the one you are trying to place on the disk. Suppose your disk was formatted with PC-DOS and you use MS-DOS. Hence, the names of the system files recorded in the boot record will be IBMBIO.COM and IBMDOS.COM; most versions of MS-DOS, however, call the equivalent files IO.SYS and MSDOS.SYS. There is also an OEM ID at the beginning of the boot record, reading IBM *N.N* or MSDOS *N.N*.

The simplest solution is to reformat the disk with the /s switch. However, if you prefer, you can read the boot record into memory with DEBUG (or another program that writes to disk sectors directly), rewrite the names of the system files and the OEM ID, and write the sector back to disk. Be sure to enter the new names in uppercase and to replace extra characters in the file name by space characters.

DOS 4.0 uses a boot record whose structure is completely different from that of earlier DOS releases. DOS 4.0 often regards hard disks formatted with other versions of DOS as invalid media, and cannot read them. Therefore, you may not be able to transfer DOS 4.0 to a hard disk without first removing the DOS partition using FDisk, and creating a new one (see FDisk).

Break

Syntax

{**1**} `break=on`
`break=off`
{**2**} `break on`
`break off`
{**3**} `break`

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Break command tells the computer how often it should check whether Ctrl-Break (in many instances Ctrl-C as well) is pressed. These key combinations tell DOS to interrupt the current activity.

If you do not issue a Break command, the default condition is off. When Break is off, DOS checks for the Break keys only when writing to the screen, the printer, or the serial ports, or when accepting input from the keyboard (see Ports, Serial and Parallel). During other types of processing (such as reading from or writing to disk, copying files, or non-DOS operations such as recalculating a spreadsheet, reformatting a word processing file, or compiling a program), DOS does not check to see whether a Break has been requested. Set Break to on if you want DOS to check more often. With Break set to on, DOS will check the Break keys any time it is active.

Procedure

Break as a Configuration Command

{1}
```
break=on
break=off
```

You can establish a default for frequency of break-checking by placing a command in your CONFIG.SYS file. This command takes the form `break=on` or `break=off`. The former command establishes more frequent break-checking. The latter command has no effect since it reiterates the default condition.

Break as a DOS Command

{2}
```
break on
break off
```

Whether or not you establish a default condition for the Break command in your CONFIG.SYS file, you can establish or change it at a DOS prompt by entering the command `break on` or `break off`. Entering the command at a DOS prompt overrides the command in the CONFIG.SYS file. You can just as easily place the `break on` command in your AUTOEXEC.BAT file as place the `break=on` command in your CONFIG.SYS file. They have the same effect.

Checking the Current Break Status

{3}
```
break
```

To determine whether break-checking is on or off, enter the Break command with no parameter. You will see one of the following messages:

```
BREAK is on

BREAK is off
```

Warnings

Several programs use Ctrl-C as a command key. In some instances, setting Break to on with such a program can have undesirable results. For example, WordStar uses Ctrl-C to page down one screen. If Break is on and you press Ctrl-C several times, you may interrupt your application instead of scrolling the screen because one of your Ctrl-Cs was passed through to DOS. Your file will be hanging in memory, unsaved. Set Break to off with such programs.

Depending on the structure of your program, you run the risk of losing data and returning to the DOS prompt if extended break-checking is on.

Tips

Some programs automatically set break-checking to on, or handle Ctrl-Break internally. In these programs, your Break command has no effect.

You may want to set Break to on when you are using a program that doesn't write to the screen or seldom sends output to the ports. For example, when running a compiler, you might want to be able to interrupt the compilation when it runs much longer than you expect it to (possibly indicating an error).

It is also helpful to turn on break-checking when using a spreadsheet program. If the program doesn't take over the Break keys, you can more easily interrupt a recalculation, which you might want to do when there is an error in a formula. It is equally useful in a database program. If you enter a command incorrectly, the result may require extensive searching, which you will want to interrupt. You can also interrupt a program that contains an infinite loop if it reads or writes to disk.

Buffers =

Syntax

{**1**} `buffers=`*`nnn`*
{**2**} `buffers=`*`nnn`* `/x`

DOS Versions

{**1**} MS-DOS 2.1 to 3.31
{**1**} PC-DOS 2.0 to 3.3
{**all**} MS-DOS 4.0
{**all**} IBM DOS 4.0

Overview

A buffer is a reserved portion of memory. The buffers created by the DOS Buffers = command are *disk buffers* and are used to speed up reading from and writing to disk. Each buffer is normally a 528-byte segment of memory. (The buffers may be larger in DOS 4.0 if you create a hard disk partition larger than 32 megabytes. See FDisk.)

B

BUFFERS=

When you request information from disk, DOS first looks in its disk buffers before going to disk. If the information is there, DOS immediately passes it on to the program or command that needs it. If not, DOS first reads the data into one of the disk buffers, then passes it on to the program. When all the buffers become full, the one used least recently is used.

This may appear to take more time than reading directly from the disk; however, in many instances the same information is needed several times. Under such circumstances, it may be more efficient to read from a buffer than to read directly from disk. After all, reading from a disk, even a very fast one, involves moving physical objects—the read/write head and/or the disk itself. Reading from a buffer only transfers information from one portion of memory to another and takes place at speeds measured in nanoseconds, whereas disks are read at speeds measured in milliseconds.

Buffering may also occur when you ask your program to write data to disk. DOS must first read a sector before writing it. If the information to be written is not exactly 512 bytes, DOS first checks if the target sector is already in a buffer. If so, the data is read from that buffer rather than from disk before it is written.

Procedure

DOS always establishes a default number of buffers. The default depends on the version of DOS and on your hardware. You change this default by adding the command `buffers=`*nnn* to your CONFIG.SYS file (see *CONFIG.SYS File*). The allowable number of buffers also depends on the version of DOS. With PC-DOS you may have from 1 to 99 buffers; with MS-DOS releases through 3.3, you may have from 2 to 255 buffers. With DOS 4.0 you can have up to 10,000 buffers if you also use one of the expanded memory boards that can be addressed by XMA2EMS.SYS (see Memory, XMA2EMS.SYS). To use this larger number of buffers, you must first load the XMA2EMS.SYS driver in your CONFIG.SYS file, configured appropriately, and then add the /x switch to the Buffers= command. You can place the buffers in expanded memory with this switch no matter how many buffers you establish.

Although setting a number of buffers is simple, determining the ideal number of buffers is considerably more complicated. Several considerations are involved:

- How much memory do you have in your system? Each buffer uses slightly over ½K. If you don't have much memory, or if the software you use requires most of your available memory, you must use a small number of buffers.
- What kind of disk drives do you have? The more drives, and the larger the drives, the more helpful it is to have more buffers, up to a point.
- How is your system organized? If you have many subdirectories, or if you nest them deeply, you may want more buffers. DOS keeps the subdirectory information in a buffer if one is available, which can speed up file searches noticeably.
- What version of DOS do you use? If you use a version of DOS 2, your system may seem sluggish if you use more than 20 buffers. If you use DOS 3.0 or later, the same problem can arise when there are more than 60 buffers. With DOS 3.3 and later, you can use the FastOpen command to speed up disk searches (see FastOpen). Thus, you needn't increase the number of buffers for this purpose.
- What kind of software do you use? Software that reads files randomly or uses overlays for part of its program code works better with more buffers. With database applications (which includes most accounting packages), you will notice improved performance when you increase the number of buffers. You may not need many buffers with software that does its own buffering.

In DOS releases before 3.3, the default number of buffers is two for diskette-based systems and three for systems including a hard disk. Unless you have very little RAM installed, this number is probably much lower than it ought to be. If you use DOS 2 and do not have a hard disk, you should probably have between ten and twenty buffers. If you have a hard disk, twenty is probably closer to the right number. If you use DOS 3.0 or later, you can probably increase these figures by ten.

DOS 3.3 and later versions include a formula that automatically establishes a default number of buffers to conform to your hardware. You might want to use it as a guide, even if you use some other release of DOS.

- For systems with no drives larger than 360K, and no more than 128K of RAM, the default number of buffers is two.

- If any disk drive is larger than 360K (this includes any hard disk, and all types of high density drives) and memory is 128K or less, the default is three.
- If memory is greater than 128K but not more than 256K, the default is five.
- If memory is greater than 256K but not more than 512K, the default is ten.
- If memory is greater than 512K, the default is fifteen.

Ultimately, you will have to determine the ideal number of buffers by trial and error. The upper limit, of course, is the amount of memory you can spare. Each additional two buffers uses an additional 1K or so of RAM. You can use the defaults established by DOS 3.3 as a lower limit.

To determine the ideal number, appraise your work habits in light of the factors outlined above. If you do a great deal of database work, or your programs read from overlay files, you probably need at least ten more buffers than the minimum suggested for your hardware.

In practice, the way to find out the ideal number of buffers is to place a minimum number of buffers in your CONFIG.SYS file, reboot, and see what happens. Thus, if you use DOS 2, have two diskette drives and 512K of memory, you might start with `buffers=15`. If you are using a 42-megabyte hard disk, DOS 3.3, and 640K of memory, start with `buffers=25`.

Periodically, after you have performed all the tasks you normally perform on your computer, edit the CONFIG.SYS file and increase the number of buffers by two or three. If at some point your system seems to be slowing down, decrease the number to its previous level. If you find your applications are becoming cramped, or if your disk drive light comes on more frequently, reduce the number. After a few days, you will find a number of buffers that works well for you.

Warnings

If your programs neither use overlays, nor read and write files sequentially (from beginning to end), increasing the number of buffers beyond a certain point will slow your system down.

If you use the /x switch in conjunction with a memory board that is not 100 percent compatible with the memory board manufactured by IBM for which the XMA2EMS.SYS driver is designed, you may find your root directory and/or file allocation table wiped out the first time you access a disk. Do not use this switch if you're not positive your board is designed for it.

Tips

If you use a disk-caching program, additional DOS buffers may not help as much as increasing the amount of cache memory; in fact, you may want to *reduce* the number of buffers at the same time.

Some commercial hard-disk backup programs function more efficiently with many buffers; others bypass the DOS buffers and do their own buffering. If you use commercial hard-disk backup software, check your manual to see which kind of program you have. It may affect the number of buffers that is ideal for your system.

Call

Syntax

`call` *`batchfile`* [*`parameters`*]

DOS Versions

MS-DOS 3.3 to 4.0
PC-DOS 3.3
IBM DOS 4.0

Overview

The Call command is a batch command that allows you to execute a batch file from another batch file without loading a secondary command processor. This capability lets you create batch files that you can call as though they were subroutines. You can then execute these files whenever needed while executing another batch file.

The problem is that this technique is useful only on a hard disk that has adequate space for numerous batch files. Even a short file takes a

minimum of 2K to 8K of storage on a hard disk (depending on the type and size of disk). You must decide whether the extra storage space you use is worth the cleaner code you can create.

Procedure

To call a second batch file from within a batch file, enter `call` followed by the name of the batch file you want to execute. If the called file contains replaceable parameter symbols, you can pass values to the called file by entering the values as part of the Call statement (see Batch Files).

Examples

Creating Batch Subroutines Suppose you normally call a utility such as SuperKey from a batch file called KEY.BAT, shown below. If you want to pass commands to SuperKey from other batch files, you needn't reproduce all the code from KEY.BAT each time you want to do so; instead, you can include a call to that batch file in the batch file that loads your application program.

KEY.BAT

PROGRAM	COMMENTS
`echo off`	Turns off display of commands
`cd\key`	Makes the SuperKey directory current
`key %1 %2 %3 %4 %5 %6 %7 %8 %9`	Passes parameters to SuperKey
`cd\`	Makes the root directory current

For example, let's say a program called APLICATN.BAT sets up and calls a program called APLICATN.EXE. APLICATN.BAT might include code such as the following:

```
echo off
call key aplicatn /ml
cd\ap
aplicatn
call key /mc
del *.bak
```

The first Call statement passes the name of your macro file (APLICATN.MAC) to SuperKey, along with the command to load it (/ml). The second Call statement sends SuperKey the command to clear the macro file from memory (/mc).

If you use earlier versions of DOS, you can accomplish the same thing using `command /c`. If you use DOS 2, add the command `exit` to the end of the file (see Command and Exit).

Creating Routines for Special Circumstances The Call command is especially useful for executing extensive routines only in particular circumstances. For example, it's good practice to back up your entire hard disk monthly and to back up your new and changed files daily. You can include code in your AUTOEXEC.BAT file to ensure that you perform these procedures on schedule, but not more often.

The code shown in HDBACKUP.BAT (see Hard Disks, Backing Up and Restoring), is one way to accomplish this. The code uses a Call statement to execute TODAYIS.BAT (shown in the ErrorLevel entry). This program places the current date in the environment where it can be known to the program (see Environment Space and Set). Then another Call statement calls a program to place the date of the previous backup in the environment, where it too can be known to the program. Additional Call statements call batch files that do a full backup or an incremental backup, depending on which is appropriate. These Call statements cannot be replaced by `command /c` since the programs they call place variables in the environment (see Command).

A Usage-Logging Program LOGON.BAT, which follows, is another program that uses Call in a manner that cannot be duplicated by `command /c`. It creates a log of people using a computer and the programs they use. Users type *logon*, followed by their first and last names, the command to start their program, and any parameters. For example,

```
logon Fred Thimsfrabble dbase
```

If the user leaves out any required parameters (first name, last name, program), the If statements direct the flow to the :message label. This begins a module that displays a message explaining the proper syntax.

LOGON.BAT

COMMANDS	COMMENTS
`@echo off`	Turns off display of commands
`prompt $p$g`	Establishes standard prompt (see Prompt)
`if %1!==! goto message`	These lines display help message if any parameters are missing
`if %2!==! goto message`	
`if %3!==! goto message`	
`echo %1 %2>>usage.log`	Places user's name in USAGE.LOG
`if %today%!==! goto execute`	If date is in environment already, does not create a new copy of NEWDATE.BAT
`echo|more|date>newdate.bat`	Creates NEWDATE.BAT with current date
`:execute`	
`copy today current.bat`	Ensures that CURRENT.BAT will place the date in the environment
`call newdate`	Calls the program that calls CURRENT.BAT to place the date in the environment
`echo %today% %3 >>usage.log`	Places the date and program name in USAGE.LOG
`echo|more|time>newtime.bat`	Creates NEWTIME.BAT with current time
`copy now current.bat`	Ensures that CURRENT.BAT will place the date in the environment
`call newtime`	Calls the program that calls CURRENT.BAT to place the date in the environment
`echo %now%>>usage.log`	Places the date and program name in USAGE.LOG
`%3 %4 %5 %6`	Calls the program, with parameters
`echo Logoff:>>usage.log`	Places the word Logoff in USAGE.LOG
`echo|more|time>newtime.bat`	Creates NEWTIME.BAT with current time
`call newtime`	Calls the program that calls CURRENT.BAT to place the date in the environment
`echo %now%>>usage.log`	Places the date and program name in USAGE.LOG
`goto :end`	Skips the help message

continues

COMMANDS	COMMENTS
`:message`	Begins the help message module
`cls`	Clears the screen
`echo ^@`	Creates a blank line on the screen
`echo ^GType LOGON followed by your first and last names, and the command echo you want to use. You may enter parameters for the command as well.`	Help message
`prompt Please log on:`	Establishes a special prompt
`:end`	

LOGON.BAT uses the techniques described in the Redirection and Pipes entry (see Redirection and Pipes) to store the current date and time in the environment. The values assigned to TODAY and NOW are then read by the program and directed into the USAGE.LOG file, along with the user's name and the program opened. The same procedure records the time the program ends. A sample USAGE.LOG file appears in Figure 8.

This program *must* use Call because of the relationship between the command processor and the environment (see Set and Environment Space). The `command /c` command, unlike Call, loads a second copy of COMMAND.COM, which creates its own environment. The values assigned to TODAY and NOW appear in that environment. However, as soon as control reverts from NEWDATE.BAT or NEWTIME.BAT to LOGON.BAT, the second copy of COMMAND.COM is unloaded and its environment is unloaded along with it. Thus, the values would disappear before LOGON.BAT could read them.

Tips

For DOS versions earlier than 3.3, you can use `command /c` in place of Call under most circumstances (see Command).

```
Lulu Reed
1-19-1988 wp
8:23:16.12
logoff:
9:42:06.87
John Hodges
1-19-1988 wp
9:50:19.19
logoff:
10:30:25.84
Fred Thimsfrabble
1-19-1988 dbase
10:35:11.84
logoff:
11:55:47.42
Lulu Reed
1-19-1988 lotus
12:58:24.52
logoff:
14:42:14.19
Jack Pollock
1-19-1988 cadd
14:49:04.28
logoff:
17:12:51.86
John Hodges
1-19-1988 pacman
17:13:43.28
logoff:
18:59:31.02
```

Figure 8: USAGE.LOG

CD

Syntax

{**1**} cd *path*
cd*path*

{**2**} cd \
cd\

{**3**} cd ..

{**4**} cd..

{**5**} cd *drive path*

{**6**} cd
cd *drive*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

On a disk with tree-structured directories, the CD (Change Directory) command makes a directory current. The longer form, ChDir, is equivalent. For a full discussion of tree-structured directories, see Directories. For a full discussion of path names, see Path Names.

Usage

The following discussion assumes that the hard disk is drive C and that the directory structure is as shown in Figure 28 in the Directories entry. For the following examples, you may want to refer to Figure 28.

Making a Directory Current

{1} `cd path`
`cdpath`

{2} `cd \`
`cd\`

Like all commands using path names, CD can use either absolute or relative path names (see Path Names). Suppose your current directory is the \WP directory on drive C and you want to make the directory called \REPORTS\MIDWEST\QTR1 the current directory. The command

```
cd \reports\midwest\qtr1
```

will make that directory current, regardless of what other directory is current, as long as you are logged onto drive C.

A relative path name may be more convenient if the directory you want to reach is a child of your current directory. Thus, if your current directory is \REPORTS and you want to make the \REPORTS\MIDWEST\QTR1 directory current, simply enter `cd midwest\qtr1`.

Notice that you do not use the leading backslash, which indicates the root directory. Similarly, if your current directory is \REPORTS\MIDWEST, all you need do to make QTR1 the current directory is to enter `cd qtr1`.

When the path name you are entering begins with the root, you may leave out the space between the command and the path name. Thus, the following two commands are identical:

```
cd \reports\midwest
cd\reports\midwest
```

When the root directory is current, you may use a relative path name that does not start with the root since all other directories are subordinate to the root. Thus, when the root is current, the following commands are all identical:

```
cd reports
cd \reports
cd\reports
```

Because the root directory's name is always \, you can make the root directory current from anywhere on the current disk by entering `cd \` or `cd\`.

Moving Up One Level

{3} `cd ..`
{4} `cd..`

You can use the double-period syntax to move to the parent of the current directory. (See Path Names.) To continue our previous example, to make the \REPORTS\MIDWEST directory current when \REPORTS\MIDWEST\QTR1 is current, simply type

```
cd ..
```

Note that when using DOS 2, you must include a space between the command and the double period. This space is no longer required in DOS 3.0 and later, making

```
cd..
```

an acceptable command.

If your subdirectories are nested several levels deep, you can use the . . directory name to move through levels. If you have a directory tree like the one shown in Figure 28 and you want to move from \REPORTS\MIDWEST\QTR1 to \REPORTS\MIDWEST\QTR3, enter the command

```
cd ..\qtr3
```

This translates as "go up one level from the current directory, and then go down one level to QTR3."

Changing Directories on Noncurrent Drives

{5} cd *drive path*

To change directories on a drive that is not current, preface the path name with a drive name. Thus, if drive A is current and you want to make the \REPORTS\MIDWEST\QTR1 current on drive C, use the command

```
cd c:\reports\midwest\qtr1
```

Just as when the drive is current, you can use a shorter path name, without the leading backslash, if the current directory on the drive you are addressing is a parent of the target directory.

Finding Out Which Directory Is Current

{6} cd

```
cd drive
```

To find out what directory is current, type CD with no parameters. In the example above, typing this command produces the response

```
C:\REPORTS\MIDWEST\QTR1
```

To find out what directory is current on a drive that is not current, include a drive name:

```
cd c:
```

Warnings

When a CD command includes a drive name, you must enter a space between the command and the parameter, but you may not have a space between the drive name and the path name. The following forms would result in error messages:

```
cdc:\reports\midwest\qtr1
cd c: \reports\midwest\qtr1
```

Tips

For related information, see Directories, MD, Path Names, Prompt, and RD.

ChCP

Syntax

{1} chcp *charset*
{2} chcp

DOS Versions

MS-DOS 3.3 to 4.0
PC-DOS 3.3
IBM DOS 4.0

Overview

ChCP (Change Code Page) is one of four configuration commands and four or five DOS commands which, together, allow you to display foreign-language characters on your screen. *Code page* is the term that IBM and Microsoft use to denote character set.

Procedure

{1} chcp *charset*

To load one of the four character sets other than your computer's defaults, first make sure that your CONFIG.SYS file has the following device drivers properly loaded:

ANSI.SYS
DISPLAY.SYS
PRINTER.SYS (if your printer can use the character sets)

See the respective entries for details. Also include a country=*nnn* command for the appropriate country (see Country=).

Next, before invoking the ChCP command, do the following:

1. Execute NLSFunc (see NLSFunc).
2. If you use a color graphics adapter or a monographics (Hercules-type) adapter, execute GrafTabl with the appropriate parameter (see GrafTabl).

3. Execute the Mode command with the CodePage Prepare function for your screen and, if necessary, for your printer (see Mode).
4. If you want to use the characters at the keyboard, load an appropriate keyboard-remapping table with the Keyb command (see Keyb).

The parameters for Mode, GrafTabl, and ChCP should all be the same.

After completing step 3, your screen should jump and the characters change to their equivalents in the character set you just loaded. (See NLSFunc for sample code to execute these steps.)

Now that your computer is ready, you can execute ChCP by entering

```
chcp charset
```

where *charset* is one of the three-digit code page numbers listed in Table 21 in the Mode entry. If you have loaded an appropriate alternative keyboard-mapping table, your keyboard will now type in the language that you have established using these commands. If you don't use a TSR that preempts the key combination, pressing Ctrl-Alt-F1 will restore the keyboard to its default state, even with the alternative character set loaded (see TSR). To return to the default character set, enter (in the United States)

```
chcp 437
```

To return to the alternative keyboard, press Ctrl-Alt-F2 and enter the original chcp command.

{2} chcp

To find out which character set is active, enter the ChCP command with no parameters.

Tips For complete information on teaching your computer to speak another language, see also Country=, COUNTRY.SYS, DISPLAY.SYS, GrafTabl, Keyb, Mode, and NSLfunc.

ChDir *See* CD

An alternative name for CD.

ChkDsk

Syntax

{1} chkdsk [*drive*]
{2} chkdsk [*drive*] /f
{3} chkdsk [*drive*] /v
{4} chkdsk [*drive*][*path*]*filename.ext*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

ChkDsk (Check Disk) reports on your disks and the files on them. It also reports on the amount of free space in your computer's memory. Depending on the parameters you use with the command, ChkDsk can do the following:

- Report on disk integrity and use
- Fix certain disk errors
- List all the file names on your disk
- Report on the efficiency of file storage

ChkDsk is an external command; using it requires having the DOS program CHKDSK.COM (or CHKDSK.EXE) on a disk in your computer. If you use DOS 2, it must be in either the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:chkdsk a:
```

If you use DOS 3.0 or later and the program is not in the current directory or on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\chkdsk a: /f
```

Although diskettes can last for a long time, many things can go wrong with them. Diskettes are sensitive to

- Grease
- Bending or creasing
- Magnetic fields
- Punctures
- Extremes of temperature

You may unthinkingly write on a disk label with a pencil or a ball-point pen, creasing the electronic medium. Or you might pick up a diskette while your hands are greasy or covered with food and get grease on the magnetic surface through the oval-shaped hole. Leaving a diskette next to your monitor, which generates a magnetic field, can damage it, as can leaving it in the sun. Even with the best of care, however, diskettes simply wear out.

Hard disks are also subject to damage and wearing out. Even when new, most hard disks have at least a few bad sectors that will not hold data properly. If you jar or move the computer while the power is on or while the read-write head is not properly parked, the head may touch the magnetic medium and scratch it. Should this happen, the scratched disk sectors will become unusable and you will lose any data that was in them.

Various practices can also damage your files, even when the disk itself is in perfect physical condition. These practices include:

- Switching disks in a drive while a program is running. Some programs keep an image in memory of the file allocation table and root directory of the current disk. If you change disks and do not inform the program, it will write files as if the old disk were still in place.
- Exiting a program improperly (for example, by rebooting the computer while a file is open). The data file will not be closed properly, and its length will not be recorded.

ChkDsk warns you when damage occurs so you can save the data that you might otherwise lose. ChkDsk checks each file on the disk against its recorded size and location as reported in the directories and file allocation table. It will report any discrepancies and give you a chance to correct them.

Entering the ChkDsk command with no parameters results in a report of the form shown in Figure 9 for all DOS releases prior to 4.0. In DOS 4.0, the report takes a slightly different form, as shown in Figure 10.

Both forms of this report have several parts that are separated by blank lines. The material in the second and third parts of the report in DOS 4.0 are combined in the second part of the report in earlier releases. In either version, the first part reports on the *volume label*. If you have labeled the disk using Format with the /v switch or using the Label command (in DOS 3), this line will display the date and time the label was created; for example,

```
Volume JK'STURBOAT created Mar 18, 1987 12:00p
```

If you did not enter a volume label, this line will read

```
Volume in drive drive has no label
```

just as in a directory display.

If you formatted the disk with DOS 4.0, and you use the DOS 4.0 version of ChkDsk, a second line of the volume label report displays a message similar to this:

```
Volume Serial Number    65E1-3AF0
```

The serial number is an eight-digit hexadecimal number, assigned automatically by the DOS 4.0 Format command (see Format).

The second part is the *disk report*, which provides information about the disk you specified as a command parameter. (If you did not specify a disk, it will report on the current disk.) This part of the report summarizes the material on the disk. The total disk space reported in the first line equals the sum of the bytes in all subsequent lines of the disk report. The

```
Volume in drive drive. . .

nnnnnn bytes total disk space
  nnnn bytes in n hidden files
   nnn bytes in n directories
nnnnnn bytes in nn user files
 nnnnn bytes in nn bad sectors
 nnnnn bytes unavailable
nnnnnn bytes available on disk

nnnnnn bytes total memory
nnnnnn bytes free
```

Figure 9: The ChkDsk report

```
Volume in drive drive. . .
Volume Serial Number   nnnn-nnnn

nnnnnn bytes total disk space
  nnnn bytes in n hidden files
   nnn bytes in n directories
nnnnnn bytes in nn user files
 nnnnn bytes in nn bad sectors
 nnnnn bytes unavailable

  nnnn bytes in each allocation unit
 nnnnn total allocation units on disk
 nnnnn available allocation units on disk

nnnnnn total bytes memory
nnnnnn bytes free
```

Figure 10: The DOS 4.0 ChkDsk report

last line of the disk report in DOS 3.3 and earlier is replaced by a separate section of the report in DOS 4.0.

The earlier releases simply report the number of bytes of additional data that the disk can hold. In DOS 4.0 this line is replaced by an *allocation unit report*, comprising three new lines. The first tells you the size of the disk's *clusters* (called *allocation units* in the report; see Disk Clusters). The second tells how many clusters are on the disk. The third tells how many clusters are available on the disk for storing data. This is a much more reliable guide to the available space on the disk than the number of bytes, because DOS allocates storage a cluster at a time. It's also much easier to estimate the total capacity of the disk, because clusters are always a multiple of half a kilobyte (512 bytes).

The final part is the *RAM report*, which indicates how much memory you have, and how much is not in use at present.

Not all of this information appears in every report. If you enter the ChkDsk command on a newly formatted standard, double-sided, double-density 5¼-inch diskette that has no bad sectors, in a system that has 512K of installed memory, the report would resemble Figure 11, in DOS releases earlier than 4.0.

Let's examine a disk report line by line. The first line shows the total storage capacity of the disk. This figure is somewhat lower than the nominal disk capacity. In the example above, the first line reads

```
362496 bytes total disk space
```

```
362496 bytes total disk space
362496 bytes available on disk

524288 bytes total memory
455651 bytes free
```

Figure 11: ChkDsk report for a 5¼" diskette

This amounts to 354K. This type of diskette has a nominal capacity of 360K. The difference comprises the DOS reserved areas – the boot record, the file allocation table, and the root directory. Note that these quantities would be obvious in the DOS 4.0 version of ChkDsk, because of the allocation unit report. If this disk were being checked by DOS 4.0, the allocation unit report would read:

```
1024 bytes in each allocation unit
354 total allocation units on disk
354 allocation units on disk
```

Since 1024 bytes is 1 kilobyte, it's obvious that the disk can hold 354K. See Diskettes, Structure of; and Disk Clusters, for further details.

If any hidden or system files are on the disk, their combined size appears in the second line of the report. (See File Attributes.) They are not reported among the user files. If the disk is bootable, this figure will include the bytes used by the DOS system files (IO.SYS and MSDOS.SYS in MS-DOS, IBMBIO.COM and IBMDOS.COM in PC-DOS). This line will not appear if the disk has no hidden or system files.

If you have assigned a volume label to a disk that is not bootable, the second line may read

```
0 bytes in 1 hidden files
```

The second line reads thus because the volume label appears among the root directory data as a file name. However, it is not actually a file, and the root directory space is not included in the report; therefore, the volume label takes up no storage space.

There may be additional hidden files if the disk contains either copy-protected software or software that you haven't yet installed. If so, these are either files used by the copy-protection scheme or files that will be

linked during the installation process. Some software packages also create hidden files that allow you to restore the data to a reformatted hard disk.

If the disk has any subdirectories, the third line of the report will show how many bytes they occupy. This will normally be a multiple of 1024 because when you write a file—including a subdirectory—DOS allots space for it in multiples of 1024 bytes. This number will be different for high capacity diskettes and hard disks. It may also be different for RAM disks. (See Disk Clusters for further information.)

If the disk contains files other than hidden or system files, their combined size in bytes appears in line 4.

The last line of the disk report indicates how many bytes on the disk are available for storing data.

The RAM report always contains two lines. The first line reports how much system (as opposed to expanded or extended) memory your computer has. (ChkDsk is not aware of other types of memory.) Unless you make changes to your hardware, this number probably will not change. The second line, the free memory report, indicates how much memory is available for your programs and the files they keep in memory.

There is always less free memory than system memory. Even if you have not loaded any programs or installed any device drivers, the three DOS programs—the system files and COMMAND.COM—always occupy some portion of memory. You will have less free memory if you have set a number of buffers or files in your CONFIG.SYS file, or installed any device drivers, or invoked any terminate-and-stay-resident programs such as GRAPHICS.COM or FASTOPEN.EXE, or commercial TSR utilities (see TSRs).

Procedure **Getting a Report**

{**1**} chkdsk [*drive*]

The ChkDsk command always produces a report such as the one described above about your disk and memory, regardless of the switches you use. If all you want is a report, enter the ChkDsk command with no switches and no arguments except the target drive. Enter ChkDsk with switches to eliminate or recover lost clusters, fix allocation errors, or extricate cross-linked files.

Eliminating Lost Clusters

{2} chkdsk [*drive*] /f
{3} chkdsk [*drive*] /v

Sometimes when you enter ChkDsk command, a message of the following form will appear:

```
nnn lost clusters found in nn chains.
Write corrections to disk (Y/N)?
```

This means that some clusters that are not part of any file named in a directory appear as allocated in the file allocation table. Moreover, all the files named in the directories have all their storage accounted for.

As mentioned, there are some circumstances under which files are not written correctly to disk. This can have several consequences, any of which can result in the appearance of these *lost clusters*. When several consecutive clusters that are not part of a file are allocated in the file allocation table, ChkDsk assumes they were once part of a single file.

These chains of lost clusters may be harmless. If a scan of all the directories shows that no files are missing, lost clusters do nothing worse than take up space; however, you're in trouble if file names that should exist are missing, or others are present that shouldn't be.

Obviously, the first thing to do is to respond to the prompt on the screen. If you invoked ChkDsk with no switches, and you press Y, the message

```
Errors found, F parameter not specified
Corrections will not be written to disk
```

will appear. If you press N, the following message appears:

```
nnnnn bytes disk space
      would be freed.
```

This means that ChkDsk would have made some corrections if you had included the /f switch in your command. There are a few other things you should do, however, before reentering the command with the /f switch.

The first step is to check your directory to see if there are any extra or missing files. If the disk has subdirectories, use a command of the form

```
chkdsk [drive] /v
```

The /v switch (for verbose), lists all the file names on your disk, with complete path and directory information. If you have many files or directories on your disk, you won't be able to read the report as it scrolls by. If this happens, redirect the output to a file on another disk or to your printer or use the Move command (see Move).

You are now ready to deal with the lost clusters. Enter the command

```
chkdsk [drive] /f
```

When you see the prompt

```
Write corrections to disk (Y/N)?
```

respond according to the following rules. If all your files are the correct length and none are missing, type an N. ChkDsk will simply adjust the file allocation table to show the formerly lost clusters as being empty.

If there is *anything* wrong with your directories, type a Y. After you do so, each chain of lost clusters will be written to a separate file, with the name FILE*nnnn*.CHK where *nnnn* is a number from 0000 to 9999. These files will all appear in the disk's root directory. You can then use the Type command to view the contents of the recovered chains which will probably be files you previously erased. If so, you can erase them again and free their space. If they are parts of files that you still need, your best bet is to replace them from a backup copy. If you don't have backup copies, give them names that help you remember what they are. If several .CHK files are parts of the same lost file, you can concatenate them with the Copy command (see Copy).

If you can't recognize these files, they are either program files or files created by programs that add formatting information, pointers, and such to their files. (Examples include dBASE III PLUS and Lotus 1-2-3.) In either case, you probably won't be able to retrieve them unless they are complete, and you won't know in what order the parts should fit together.

A root directory of a disk can hold only a fixed number of file names. (The number depends on the type of disk.) You must make more room if the directory has space for fewer files than ChkDsk needs to recover. Copy the recovered files you want to save to another disk, then erase the recovered files. Next, issue the chkdsk [*drive*] /f command again and respond to the prompt by typing Y. Repeat these steps until there are no other chains of lost clusters.

Checking File Fragmentation

{4} chkdsk [*drive*][*path*]*filename.ext*

When DOS writes files to disk, it does not always write them in consecutive disk clusters. (See File Allocation Table for more information on this phenomenon.) DOS needs more time to read files that are broken up into several chains on disk; this is especially noticeable on a hard disk where many files are often updated. A great deal of fragmentation can slow down your hard disk considerably.

To find out both how many files are not contiguous and how many fragments they contain, use a command of the form

chkdsk [*drive*][*path*]*filename.ext*

Wildcard patterns are acceptable. Use the file name *.* to get a report on all files in a directory. If the file you named is not laid out consecutively, this version of ChkDsk displays a report of the form

```
C:\PATH\FILENAME.EXT
  Contains n non-contiguous blocks.
```

in addition to the standard report. If it is contiguous, you will see the message

```
C:\PATH\FILENAME.EXT
  All specified files are contiguous.
```

This form of the ChkDsk command reports on only one subdirectory. To get a full report on your disk, you must repeat the command for each directory or use a utility program that creates such reports.

You're probably wondering what you should do about fragmentation. If the disk is a diskette, copy all the files to a blank, formatted diskette with either of these commands:

```
copy [sourcedrive]*.* [targetdrive] /v
xcopy sourcedrive targetdrive /v
```

You can then erase the original and recopy the files or save the original as a backup.

If the disk is a hard disk, the problem is more complicated. You can buy some *disk-optimizing software* and follow the directions carefully. If you don't want to spend the money, the only alternative is to back up the entire hard disk, format it, and restore it.

It's relatively easy to unfragment files; allocation errors are more difficult to correct.

Allocation Errors Something is seriously wrong with your file allocation table if you see either of the following messages when you run ChkDsk:

```
B:\FILENAME.EXT
  Allocation error, size adjusted.
```

or

```
B:\FILENAME.EXT
  Invalid cluster, file truncated.
```

The first message means the file size reported in the directory is larger than the number of clusters allocated to it in the file allocation table. The second message means that the file allocation table shows more clusters assigned to the file than the directory indicates.

The next step depends on whether you used the /f switch. If you did not, ChkDsk will not make any changes; however, you should examine the specified file to see that it is complete.

If you used the /f switch, DOS adjusts the directory to match the file allocation table. If this happens, you may have lost data. To find out, examine the file whose length was changed. Use the Type command, DEBUG, or a word processor, as appropriate. If you haven't lost data, you needn't proceed further.

If you have lost data, ChkDsk /f may have written the missing data to new .CHK files and you'll be able to recover it. If it hasn't, you may have to read the file's disk sectors directly into memory using DEBUG and then write them out to a new file (see DEBUG). The catch is that you must figure out which disk sectors the file occupied. At this point, not even the file allocation table will help because it's been altered. This is one of the points at which DOS cannot proceed further. The only alternative is a

disk-exploration or unerase program. Ideally, the program will let you search the erased file space for data that was in the missing part of the file.

Cross-Linked Files Sometimes ChkDsk displays a message such as

```
B:\FILE1.EXT
  Is cross-linked on cluster nnn
B:\FILE2.EXT
  Is cross-linked on cluster nnn
```

This means that, according to the file allocation table, two different files occupy the same cluster. If you see these messages, one of the files is surely damaged.

When two files are cross-linked, DOS arbitrarily assigns the cluster in question to one of the files. You should immediately copy both files to another disk and examine them. One file will be intact, but the other will probably both have some irrelevant data and be missing a part. If you're lucky, the missing part will be in a chain of lost clusters. ChkDsk will have told you if there are any on the disk. If there are, repeat the command with the /f switch to see if you can find the missing part.

Of course, it's easier to recover a text file where you can recognize the contents. If the file is some other type, it may be better to copy the file from one of your backups.

Damage That ChkDsk Cannot Handle There is some damage that ChkDsk cannot repair. If you see the message

```
Disk error reading drive drive
Abort, Retry, Fail?
```

ChkDsk may have found a bad sector either in one of the DOS reserved areas—either the boot record, file allocation table, or root directory—in the file it's trying to read, or in the subdirectory holding the file. Type R for Retry. If the same message appears after several attempts, the disk is probably unusable. If you see the message

```
File allocation table bad, drive drive
Abort, Retry, Fail?
```

you will probably need to use the Recover command to retrieve any data from the disk, and reformat the disk (see Recover). If you see the message

```
Sector not found error
Abort, Retry, Fail?
```

while attempting to read a directory, ChkDsk has found a bad sector in a file it's trying to read. If it's in one of the reserved areas, you may need to use commercial file-recovery or disk-repair software to read anything from the disk.

In DOS releases before 3.3, the prompt

```
Abort, Retry, Fail?
```

has been replaced by

```
Abort, Retry, Ignore?
```

In any of the above instances, type A for Abort before proceeding further.

Warnings **Make backup copies of all your files, early and often!!!**

ChkDsk will not work on a network drive or on a drive that is affected by an Assign, Join, or Subst statement.

Locking out the bad sectors doesn't always make a disk safe to use, and bad sectors tend to expand. If a diskette has bad sectors, back up the files and reformat it. If the bad sectors don't disappear after reformatting, throw the disk away or return it to the manufacturer for a replacement.

If the number of bad sectors on your hard disk starts increasing regularly, do a complete backup immediately and have your hard disk repaired or replaced.

Tips **Make backup copies of all your files, early and often!!!**

Date-Stamping Your Disks If you label a disk when you create it, you can use ChkDsk to see how long it's been in use. (If you change the label, the date changes, just as it does when you modify a file.) ChkDsk is the only command that displays the date and time a volume label was entered. If you format your disks with DOS 4.0, ChkDsk always displays the date and time of creation, even if you don't enter a volume label.

(Of course, this only works if you set the date and time when you boot up, or if your computer does this automatically; see Date and Time.)

A Complete List of Files As mentioned previously, the /v switch causes ChkDsk to display a complete list of all the file names on your disk. This list is much more complete and easier to read than the display created by the Tree command.

A sample of the output from `chkdsk /v` appears in Figure 12. Notice that the names of hidden and system files appear along with those of normal files. Notice also that the file names appear in the exact order in which they appear in the directories. Thus, the \DBASE\FLASH subdirectory of \DBASE, and all the files on it, appear before the final three files in the \DBASE directory.

If you have a hard disk, you can use this command to help you find files on it. Redirect the output to a file, then use the Find filter to search that file for the file name. For example,

```
chkdsk /v> harddisk.dir
find JUNE87.DAT harddisk.dir
```

might produce the result

```
-------- HARDDISK.DIR
     C:\BUSINESS\SALES\JUNE87.DAT
```

thus letting you see exactly which directory contains the file.

Increasing Free Memory If your free memory is less than the amount required to run your programs, you must modify your setup. You can do one of the following things:

- Lower the number of buffers or files in your CONFIG.SYS file
- Remove some device drivers from your CONFIG.SYS file
- Unload some TSRs

```
Volume JK'STurboAT created Mar 18, 1987 12:00p
Directory C:\
      C:\IBMBIO.COM
      C:\IBMDOS.COM
      C:\JK'STURB.OAT
      C:\COMMAND.COM
      C:\ANSI.SYS
      C:\ANSI43.SYS
      C:\DM.SYS
      C:\DRIVER.SYS
      C:\MSMOUSE.SYS
      C:\RAMDRIVE.SYS
      C:\X2CDRIVE.SYS
      C:\EMM.SYS
      C:\AUTOEXEC.BAT
      C:\CAROUSEL.OPT
      C:\LASTBKUP
      C:\MIRROR.FIL
      C:\MIRORSAV.FIL
      C:\SPOOL.COM
      C:\TREEINFO.NCD
Directory C:\DBASE
      C:\DBASE\ASSIST.HLP
      C:\DBASE\DBASE.COM
      C:\DBASE\DBASE.LD1
      C:\DBASE\DBASE.MSG
      C:\DBASE\DBASE.OVL
      C:\DBASE\DBASEINL.OVL
      C:\DBASE\HELP.DBS
Directory C:\DBASE\FLASH
      C:\DBASE\FLASH\FL.COM
      C:\DBASE\FLASH\FLASHUP.COM
      C:\DBASE\FLASH\FLCLIP.OBJ
      C:\DBASE\FLASH\FLDEFALT.SYS
      C:\DBASE\FLASH\FLEDITOR.000
      C:\DBASE\FLASH\FLEDITOR.CHN
      C:\DBASE\FLASH\FLGEN.DB2
      C:\DBASE\FLASH\FLGEN.DB3
      C:\DBASE\FLASH\FLHELP
      C:\DBASE\FLASH\FLHLPWIN
      C:\DBASE\FLASH\FLINSTAL.CHN
      C:\DBASE\FLASH\FLMAIN.CHN
      C:\DBASE\FLASH\FLMAKDB2.000
      C:\DBASE\FLASH\FLMAKDB2.CHN
      C:\DBASE\FLASH\FLMAKDB3.000
      C:\DBASE\FLASH\FLMAKDB3.CHN
      C:\DBASE\FLASH\FLMAKHLP
      C:\DBASE\FLASH\FLMAKPRO
      C:\DBASE\FLASH\FLMAKWIN.000
      C:\DBASE\FLASH\FLMAKWIN.CHN
      C:\DBASE\FLASH\FLMENUS.WIN
      C:\DBASE\FLASH\FLSCREEN
      C:\DBASE\CONFIG.DB
      C:\DBASE\DBFIX.EXE
      C:\DBASE\DBLOG.FIX
      C:\DBASE\DBLOGM.FIX
      C:\DBASE\DBASE.EXE
Directory C:\DICT
```

Figure 12: Partial output from the ChkDsk /v command

If you make any changes to your CONFIG.SYS file, you must reboot your computer to make the changes effective.

For related information, see Attrib, Boot Record, Directories, Disk Clusters, Disk Sectors, File Allocation Table, File Attributes, Find, Format, Label, Recover, and Vol.

Cls

Syntax

```
cls
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview The Cls command clears the screen.

Tips Some commercial TSR programs cause the screen to clear to black on a color monitor, even if you have changed your screen and character color (with a utility program or ANSI escape sequences). When such programs are loaded, the border will change to black as well, even if you have established a color for it. If you have ANSI.SYS loaded, you can clear the screen to your established colors, without disturbing the border, by executing the command `echo ←[2J` in a batch file (see ANSI.SYS). You could, for example, create a file called CS.BAT (for Clear Screen), which contains only this command. Then you can use Cs instead of Cls to clear the screen.

Command

Syntax

{1} command
{2} command /c *command*
{3} command [*drive*][*path*] /p
{4} command /e:*nnnn*

DOS Versions

{1}{2}{3}	MS-DOS 2.1 to 3.1
{all}	MS-DOS 3.2 to 4.0
{1}	PC-DOS 1.0 to 1.1
{1}{2}{3}	PC-DOS 2.0 to 3.1
{all}	PC-DOS 3.2 to 3.3
{all}	IBM DOS 4.0

Overview

The Command command loads a second copy of COMMAND.COM, DOS's command processor, into memory. This copy is called a *secondary command processor*.

Because COMMAND.COM is needed to run all other programs, at least part of COMMAND.COM's code is always resident in memory. When you load a second copy, you can run an additional program without unloading the program already present.

A secondary command processor creates its own environment space (see Environment Space). Anything in the environment of the first command processor will still be available to the secondary command processor, but anything you place in the environment after loading the secondary command processor will disappear when you remove it from memory.

Procedure

Although Command is an external command, it is not subject to the same restrictions as other external commands. Since you cannot run your computer without it, DOS treats it in a special way. Unless you direct DOS to find it elsewhere with the Shell = command, it must always be in the root directory of the boot disk (see Shell =). In addition, DOS places a message to itself in the environment to remind itself where COMMAND.COM is located. This reminder takes the form

COMSPEC=*drive*\[*path*\]COMMAND.COM

DOS must always know where to find COMMAND.COM because some programs need the memory that COMMAND.COM uses, and they overwrite it. When you exit from such a program, DOS must return COMMAND.COM to memory before it can proceed. To ensure that it does not alter the operating environment by accident, DOS always reads COMMAND.COM from the same copy of the file on the same disk from which it was originally loaded. This is why you may sometimes see the message

```
Insert COMMAND.COM disk in drive A:
Then press any key when ready . . .
```

Loading the Command Processor

{**1**} command

The Command command, when entered with no parameters, simply loads a secondary command processor, which enables you to issue DOS commands (including those that run programs) while a program is in memory. Programs that provide access to DOS while they are running most commonly do so by loading a secondary command processor. When you load a secondary command processor with this syntax, you return to the original environment—or to the program already in memory—by entering the command exit.

Executing a Secondary Batch File

{**2**} command /c *command*

The /c switch loads a secondary command processor, executes the command *command*, and exits automatically to the environment of the first command processor after the command has been carried out. This syntax is used in batch files to execute a second batch file and return to the original batch file after the second file has finished executing. In DOS 2, batch files called by command /c must end with the Exit command (see Exit).

If you use DOS 3.3, you can use the Call command instead of command /c (see Call).

Replacing the Original Command Processor

{3} command [*drive*][*path*] /p

The /p (for permanent) switch loads a permanent copy of COMMAND-.COM. You cannot return from a permanent copy to a previous one with the Exit command. However, anything you placed in the environment of the original command processor remains available.

As soon as COMMAND.COM is loaded, it makes the directory from which it was loaded current. Next, it executes the AUTOEXEC.BAT file, if one is present in the same directory (see AUTOEXEC.BAT File).

Although you lose access to the original command processor when you use the command /p command, it is not removed from memory. Rather, the secondary command processor is loaded in memory above whatever is already in memory. This leaves you with about 4K less usable memory.

If you include a drive name and path name, you can tell DOS to load COMMAND.COM from a source other than the one from which the original copy was loaded. If you do, however, be sure that it is the same version as the original. If it isn't, you will see the message Invalid COMMAND.COM as well as a prompt to insert a disk with the correct version in the specified drive.

The /p switch is useful for telling DOS to load COMMAND.COM from a RAM disk when you use DOS 2. With DOS 3 or later, you can accomplish the same thing more easily with the Set command.

Setting the Size of the Environment

{4} command /e:*nnnnn*

COMMAND.COM always creates an environment for itself (see Environment Space). This space stores the directive that tells DOS where to find COMMAND.COM, the path, the prompt, and other information that you place there with the Set command. In addition, many programs place information in this area for their own use.

The environment is finite in size. If you try to place more in it than it will hold, you will see the message Out of environment space and you will not be able to place anything else in it.

Note that under some circumstances, DOS can increase the size of the environment as needed. If you have not loaded any TSRs, DOS increases the size of the environment in 127-byte increments as needed. However, once a TSR is loaded, DOS can no longer increase the size of the environment.

To change the default size of the environment, use the /e: switch, followed by the size of the environment in bytes. The number must be a multiple of 16 between 160 and 32,768. An environment of 960 to 1088 bytes is large enough to cover most contingencies, even with a long path and prompt.

In DOS 4.0, you can use the /msg switch to keep the text of DOS messages in memory. This is useful on a diskette-based system in which the diskette with COMMAND.COM may not be in a drive when a message is generated. Without the /msg switch, only message numbers will be displayed, and you'll have to look up their meaning in the DOS manual. You must use the /p switch with the /msg switch. It's more efficient, however, to use the Shell= command to keep the messages in memory. See Shell= for details.

Examples

You can find examples of how to use Command in the AUTOEXEC.BAT File entry. One example includes the command to call a hard-disk backup batch file from an AUTOEXEC.BAT file:

```
command /c hdbackup
```

Another example shows how to use `command /p` to tell DOS to find COMMAND.COM on a RAM disk in a diskette-based system. The advantage of doing so is that the RAM disk cannot be removed. Therefore, when DOS needs to reload COMMAND.COM, there is no chance that you will have removed the disk containing it from a drive.

Tips

If you use DOS 3.1 or later, you can save some memory by using the Shell= command in your CONFIG.SYS file, instead of `command /e:`, to specify the size of the environment. Shell= does not load a secondary command processor.

In DOS 3.0 and later, you can direct DOS to use a different copy of COMMAND.COM by entering the command

```
set comspec=drive\path\command.com
```

Command Characters

Overview Several characters have special meaning to DOS. They are used as parts of commands and tell DOS to do something. You cannot use these characters as parts of file names or directory names. The characters and their meanings appear in Table 5.

TABLE 5 **DOS Command Characters**

CHARACTER	PURPOSE
"	Specify text for the Find command to find (see Find)
*	Wildcard character for file names (see Wildcard Characters)
+	Concatenation operator used with Copy command (see Copy)
,	Separator used in some command lines
.	Separator placed between file name and extension
/	Switch character used to denote a switch used with a command[1]
:	Drive name indicator
;	Separator between path names in a path command
<	Redirection operator (see Redirection and Pipes)
=	Assignment operator for environment strings and configuration commands (see Set, CONFIG.SYS File)
>	Redirection operator (see Redirection and Pipes)
?	Wildcard character for file names (see Wildcard Characters)
\	Separator between directories in a path name
|	Pipe operator (see Redirection and Pipes)

[1]*Some MS-DOS versions, particularly early releases, use the hyphen as the switch character. In these versions, the / character is a command character that separates directory names in a path. DOS 2 includes an undocumented command, Switchar=, which allows you to specify the switch character in your CONFIG.SYS file (see Switchar=).*

Command Keys

Overview

When you are at the command level (that is, at a DOS prompt), some of the keys and key combinations on the keyboard have special meaning to DOS. These keys and key combinations affect the way the computer behaves or the way the command line appears: some of them allow you to edit the command line. They have the same effect in the DOS programs EDLIN and DEBUG that they have at the DOS prompt. These keys and their effects are summarized in Table 6.

TABLE 6

DOS Command Keys

KEY	EFFECT
Editing Keys	
F1	Enters the character that is in the template at the current distance from the prompt into the current command line.
F2 *c*	Copies all characters from the template from the present position of the cursor to the character *c* to the new command line.
F3	Copies all characters in the template from the current cursor position to the end of the template to the new command line.
F4 *c*	Copies all characters in the template after the character *c* to the current command line.
F5	Whatever you enter on the command line before you press F5 becomes the new template. Pressing F5 places an @ character on the screen immediately after the text you entered and moves the cursor to the following line, about three quarters of the way across. You can then enter a new command, using whatever you entered before pressing F5 as the template.
F6	Inserts an end-of-file marker (Ctrl-Z).
F7	Inserts an ASCII 0 (Ctrl-@, or NUL).
Ins	Assuming you are using a template, Ins allows you to add characters at the current cursor position on the current command line. Any characters that you then enter from the template using keys F1 through F4 will begin with the character that was in the position in the template at which you pressed Ins.
Del	Each press of the Del key skips a character in the template on the current command line.

Table 6 continues

KEY	EFFECT
Escape	Cancels the current command line without affecting the template, enters a \ character at the point where Esc was pressed, and moves the cursor to the next line, below the entry point for the previous line. You can then enter a new command, using the previous template if you wish.
Ctrl-C	If Return has not been pressed, cancels the command on the current command line without affecting the template. If Return has been pressed, may cancel the current command, depending on the status of the Break command (see Break).
Backspace	Deletes the last character on the command line, moving the cursor left.
Tab	Moves the cursor to the next tab stop on the current line (tab stops are every eight screen columns).
Ctrl-H	Same effect as Tab.
Ctrl-I	Same effect as Backspace.
→	Same effect as F1.
←	Same effect as Backspace.
Ctrl-Break	Same effect as Ctrl-C.

Other Keys

Return	Enters the command on the command line and places it in the template buffer.
Ctrl-C	Depending on the status of the Break command, may interrupt the command currently being executed (see Break).
Ctrl-Break	Same effect as Ctrl-C.
Ctrl-P	Copies what appears on the screen to the printer, as it appears, character by character. Characters you type will appear on the printer as soon as they are typed.
Ctrl-PrtSc	Same effect as Ctrl-P.
Ctrl-S	Stops the screen display from scrolling. Pressing any other key resumes scrolling.
Ctrl-Num Lock	Same effect as Ctrl-S.
Shift-PrtSc	Copies the entire contents of the screen to the printer if the printer is turned on. If the printer is off or none is connected, may lock up the computer.

Procedure

Table 6 is largely self-explanatory. The exception is using the command template. Each time you enter a command at a DOS prompt, or in EDLIN or DEBUG, DOS stores what you have entered in a template buffer. This buffer holds only the last command that you entered. You can use the editing keys either to reenter the previous command unaltered or to enter parts of the previous command to create a new command. Pressing Return at a DOS prompt without entering a command empties the buffer.

Think of the template as storing characters in a series of columns, beginning with a hypothetical left margin at the point where the cursor appears after the prompt. Each character you enter from the template is the character in the same column as the current cursor position on the current command line. The only way to change that is with the Ins and Del keys. Thus, if the previous command was `md directry` and you entered *c*, the next character that you could enter from the template would be the *d* from md. If, instead of *c*, you entered `del`, the next character from the template would be the *d* in `directry`. The command would then read `deldirectry`.

If you want the command to read `del directry`, you must first to use the Ins key to enter a space.

Examples

Of the keys that edit the command line, only F1, F3, Ins, Del, and the arrow keys seem useful. These keys allow you to correct an entry with an error and reenter it. For example, if you enter

```
copy dumbfile.exit d:smrtfile.ext
```

when you meant to enter

```
copy dumbfile.ext d:smrtfile.ext
```

you can hold down either the F1 key or the right-arrow key until the command line reads

```
copy dumbfile.ex
```

Then press Del to delete the *i*, and press F3 to reenter the rest of the line. This is much simpler than trying to remember the effects of the F2 and F4 keys.

You can also use these keys when you want to issue several related commands. For example, if you want to delete the original file that you copied, type `del` followed by a space (to replace the *y*), hold down the F1 key until the file name appears, then press Return.

Command Processing Order

Overview

When you enter a command on the command line, DOS takes certain actions in a specific order. First, it searches COMMAND.COM to see if you have entered an internal command. If it is not an internal command, DOS looks for a program to execute.

If your command does not include a drive name or path name, DOS looks on the current directory for a .COM file with the name you entered. If it finds none, DOS then looks for an .EXE file; if it doesn't find an .EXE file, it looks for a .BAT file (a batch file).

If your command includes a drive name but not a path name, DOS looks for a .COM, .EXE, or .BAT file, in that order, in the default directory of the specified disk. If you include a path name, it looks in the specified directory for files of these types, on the default drive if you did not include a drive name, or on the specified drive if you did.

If DOS does not find an executable file after searching the specified or default directory, it then begins searching the directories listed in your search path. It searches the directories in the order specified in the Path statement, regardless of the position of the directories on disk or the position of the directory names in the root directory. Only after DOS has completed all these steps does it return a `Bad command or file name` message.

In DOS 4.0, you can override this order using the `/path` parameter to the Append command (see Append).

Tips

You can speed up the search by increasing the number of buffers specified in the CONFIG.SYS file (see Buffers=). A portion of the directory tree is placed in the buffers each time you enter a command that requires DOS to search for a file. The search takes place more quickly when more of the tree is in the buffer.

If you use DOS 3.3, you can further increase the speed of the search by using the FastOpen command (see FastOpen).

One implication of DOS's search pattern is that an .EXE file will never be executed if there is a .COM file of the same name in the same directory. Similarly, the batch file will never be executed if there is either a .COM file or an .EXE file of the same name as a batch file in a given directory. However, you can take advantage of the search pattern to have a batch file load a program of the same name. If you put your \BATCH directory at or near the beginning of the search path, you can create a file called SC5.BAT to set up and execute SuperCalc 5 (SC5.EXE) on another directory (see Directories and Path).

Com*n*

The reserved name for serial ports, where *n* is a 1 or 2 or, in DOS 3.3 or later, a number from 1 to 4 (see Device Names).

Comp

Syntax

{1} comp
{2} comp [*drive*][*path*\]*file1.ext* [*drive*][*path*\]*file2.ext*

DOS Versions

PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

Comp compares two files. If the files are the same, Comp tells you so; if they are not the same, it displays up to ten differences and then quits. It will not compare files of different lengths.

Procedure

Comp is an external command; using it requires having the DOS program COMP.COM on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:comp report.txt a:
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\comp \reports\sales.doc a:sales.bak
```

{1} comp

If you enter the Comp command with no parameters, DOS prompts you with `Enter primary file name`. You must enter either a file name, a drive name, a path name, or any combination of the three. If what you enter

does not include a file name, DOS assumes that you mean all files on the directory specified. To specify the default drive and directory, enter at least the drive name.

If you do not enter a file name, DOS attempts to compare all files on the drive and directory you specify with files whose names match the name you enter in response to the second prompt, which is `Enter 2nd file name or drive id`. If the files have the same name but are on different drives, you need enter only the drive name at this prompt. You may, however, specify a drive, a directory, or both. If you do not specify a file name for either the primary or the secondary file, DOS attempts to compare all files on the current drive and directory with files of the same name on the other drive and/or directory.

If the files are not the same length, DOS displays the messages

```
Files are different sizes

Compare more files (Y/N)?
```

If they are the same size, DOS compares the files. If they are the same, the messages

```
Files compare ok

Compare more files (Y/N)?
```

appear.

Some programs write all files in multiples of 128 bytes, showing the true end of the file with an end-of-file mark (Ctrl-Z). To deal with this, DOS checks for an end-of-file mark during the comparison. If one or both of the compared files does not have one, DOS displays the message `Eof mark not found`. If you do not see this message and Comp shows differences at the end of the file, DOS may be looking for the end-of-file mark. In this case, the differences are not significant.

If the files are different, DOS tells you the location of the mismatch as an offset (in hexadecimal) from the beginning of the file. It also displays the hexadecimal value of the byte in each file at that location. The message takes the form

```
Compare error at OFFSET nn
File 1 = nn

File 2 = nn
```

If ten bytes are found that do not match, the messages

```
10 Mismatches - ending compare

Compare more files (Y/N)?
```

appear.

You can use wildcard characters in either the first or the second file name, though there are some tricks in doing so successfully. These tricks are illustrated in the Examples section.

{2} comp [*drive*][*path*\]*filename.ext* [*drive*][*path*\]*filename.ext*

If you include the first and second files to compare as parameters to the Comp command, you will not be prompted for them. Comparison will begin immediately. All the same rules apply to this form of the command.

Examples

The simplest way to use Comp is to compare two files by giving the name of each. Commands such as any of the following will compare only the two files specified:

```
comp sales.doc b:sales.doc
comp sales.doc sales.bak
comp sales.doc c:\reports\sales.doc
```

To compare a file on a different directory with a file of the same name on the current directory, enter the name and location of the other file:

```
comp c:\reports\sales.doc
```

This command will compare SALES.DOC in the \REPORTS directory of drive C with SALES.DOC in the current directory. The command is the same as

```
comp sales.doc c:\reports\sales.doc
```

Things begin to get complicated if you use wildcards in your file names. Be sure that your two file specifications cannot be construed as referring to the same directory. If they can be construed in this way, DOS compares every file matching the specification to itself, as Figure 13 shows.

If you type *.* for the primary file specification, and only a drive or directory for the secondary specification, DOS looks for a file name on the second directory matching each file name on the primary directory. This is a good way to find out whether all the files on the primary directory were backed up properly.

Tips

The Comp command is especially useful if you suspect that one of your program files has become corrupted. If one of your programs is misbehaving, compare the copy you are using with the copy on the distribution diskette. If the files do not match, copy the file from the distribution

```
C:\KEY>comp *.mac ?w.mac

C:MAP.MAC and C:MW.MAC

C:MW.MAC - File not found

C:SHOWCAL.MAC and C:SW.MAC

C:SW.MAC - File not found

C:FKEYPAT.MAC and C:FW.MAC

Files are different sizes

C:EV.MAC and C:EW.MAC

C:EW.MAC - File not found

C:START.MAC and C:SW.MAC

C:SW.MAC - File not found

C:SXTT.MAC and C:SW.MAC

C:SW.MAC - File not found

C:AW.MAC and C:AW.MAC

Files compare ok

C:FW.MAC and C:FW.MAC

Files compare ok

Compare more files (Y/N)?n

C:\KEY>_
```

Figure 13: Sample output from Comp command

diskette to your working disk. Be aware, however, that some programs rewrite their code when you install them. If you have this type of program, you should have made a backup copy of the program as soon as you finished installing it.

The MS-DOS file-comparison command is FC. If you use MS-DOS rather than PC-DOS, see FC.

CON. *See* Device Names, Copy CON

CON is the DOS device name for the console (that is, the keyboard and screen).

CONFIG.SYS File

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The CONFIG.SYS file is a special file you can create to accomplish any or all of three purposes:

- To load device drivers for hardware devices that your computer may not know about
- To change the default values of a number of operating system parameters
- To install TSRs in memory (see TSRs)

The device drivers that come with DOS are

VDISK.SYS (in PC-DOS 3.0 and later, and some versions of MS-DOS)
RAMDRIVE.SYS (in most versions of MS-DOS 3.0 and later)
ANSI.SYS
DRIVER.SYS (in DOS 3.2 and later)
XMAEM.SYS and XMA2EMS.SYS (in DOS 4.0)
DISPLAY.SYS (in DOS 3.3 and later)
PRINTER.SYS (in DOS 3.3 and later)

You can load any of these drivers by placing the appropriate Device= command in your CONFIG.SYS file (see Device=).

You may need to load other device drivers by placing Device= commands in the CONFIG.SYS file as well. Hardware such as external disk drives, expanded memory boards, mice, or real-time clocks, may require device drivers. In addition, some software requires you to install a device driver with the Device= command.

The commands that change default values are

Break=
Buffers=
Country=
Driveparm=
FCBs=
Files=
Lastdrive=
Shell=

Not all of these commands are available in all releases of DOS. See the various commands for the versions of DOS in which they are included.

In DOS 4.0 you can install TSRs using the Install= command in your CONFIG.SYS file, although you can also install them by other means (See Install=).

Before loading COMMAND.COM, DOS looks for a CONFIG.SYS file on the root directory of the boot disk. If one is present, DOS reads it, attempts to load any device drivers listed in it, and establishes the defaults set by any other commands in the file.

If you need different configurations for different purposes, your procedure will depend on your hardware. For example, you can create a series of bootable diskettes, each of which has a different CONFIG.SYS file. Or, if you have a hard disk system, you can use the techniques illustrated in SWAPNAME.BAT, USEAUTO.BAT, and NOAUTO.BAT in the Batch Files entry. However, you can also use separate diskettes if you don't mind booting your system from diskettes.

Procedure

Creating the File To create a CONFIG.SYS file, use any word processor or text editor that creates ASCII (nondocument) files. You can also use EDLIN or the Copy CON command (see EDLIN and Copy CON). Place each command on a separate line, beginning at the left margin.

The commands in the file will take effect the next time you reboot your computer. There is no way to unload the devices installed by CONFIG.SYS or to change the values set by it (except for Break) other than rebooting your system.

If you load device drivers, either the device driver files must be on the root directory of the boot disk or the complete path to them must be specified.

Order of Commands Many programs and pieces of hardware that come with device drivers ask you to load their driver first—before any other commands in your CONFIG.SYS file. Obviously, this is impossible if you have several such drivers. Hence, as a rule of thumb, place the CONFIG.SYS commands in the following order:

1. Device drivers for physical disks
2. Device drivers for memory boards
3. Other hardware device drivers
4. Software device drivers
5. Configuration commands

If you have several drivers of the same type (for example, drivers for several memory boards), you will have to experiment to determine the order that works best.

The order in which disk device drivers are loaded determines the drive name of the devices they control. Letters are assigned beginning with the letter after the one that designates your last internal physical drive. Thus, if you have only two diskette drives, any additional disk device drivers will be assigned letters beginning with C. If you have a hard disk, (generally drive C), the disk controlled by your first disk device driver will be D. If you have two hard disks, or two hard disk partitions, it will be E.

Similarly, if you have more than one memory board for which you need a device driver, the order in which you load the device drivers determines the memory addresses of the boards. Some boards can configure themselves automatically to use the next available memory address. With others, you must choose their memory address by setting jumpers or DIP switches. Generally, things will go more smoothly if you load the drivers for the latter type of board first and let the self-configuring boards find their own addresses above the other boards (see Memory).

Examples

What you place in your CONFIG.SYS file depends on your hardware configuration more than on anything else. Generally, the more disk drives you have (and the larger they are), the more buffers you will want; on the other hand, if you have less memory, you will want to give up less of it to configuration commands. For a two-diskette system with 256K of RAM, your CONFIG.SYS file might read:

```
files=20
buffers=10
break=on
```

On my own system, I use several different CONFIG.SYS files for different purposes. I keep them all on the root directory of my hard disk, each with a different extension. To use a different configuration, I rename the current CONFIG.SYS file to CONFIG, then rename the one I want to use to CONFIG.SYS. The program that follows shows the CONFIG.SYS file I use most frequently. This file is designed for a PC AT system with two expanded memory boards of different types, one 1.2-megabyte diskette drive, one 360K diskette drive, a 30-megabyte hard disk, a mouse, and 640K of system memory, running under MS-DOS 3.2. X2CDRIVE.SYS establishes up to four nonvolatile RAM disks designated D through G. DRIVER.SYS makes it possible to copy files from one 1.2-megabyte diskette to another. Without this driver, which creates a logical drive H that coexists in the same space with drive A, the only way to copy files on such a disk is with the DiskCopy command.

A CONFIG.SYS File

COMMANDS	COMMENTS
`device=emm.sys 258,00`	Driver for expanded memory board
`device=x2cemm.sys`	Driver for second expanded memory board
`device=x2cdrive.sys 0,448,ne;1,96,nre`	Nonvolatile RAM disk driver
`device=driver.sys /d:0 /f:1 /c`	Driver for second logical 1.2-megabyte drive
`device=dm.sys 35 ig/b p l`	Driver for Display Master, an EGA control program

continues

COMMANDS	COMMENTS
device=msmouse.sys /1	Mouse driver
shell=command.com /p /e:1088	Expands environment to 1088 bytes
files=20	Allows up to twenty files to be open
buffers=30	Establishes thirty memory buffers
lastdrive=j	Allows the computer to recognize up to ten logical disk drives
break=on	Establishes frequent break-checking

When all the device drivers are loaded, the environment is expanded, and space is reserved for the buffers and file handles (see Files=). My computer then has about 70K less usable memory than it otherwise would have. Some programs, however, need all the memory they can get. For example, when using DESQview, a windowed multitasking environment, I need to give it more memory. Therefore, I use the substitute CONFIG.SYS file that follows.

A CONFIG.SYS File for Use with DESQview

COMMANDS	COMMENTS
device=qext.sys	Device driver for DESQview
device=emm.sys 258,00	Driver for expanded memory board
device=x2cemm.sys	Driver for second expanded memory board
device=msmouse.sys /1	Mouse driver
shell=command.com /p /e:256	Expands environment space
files=20	Allows up to twenty files to be open
buffers=20	Establishes twenty memory buffers
break=on	Establishes frequent break-checking

Notice in this program that the RAM disk driver, X2CDRIVE.SYS, and DRIVER.SYS have been removed. Because this eliminates five drive letters, I don't need the lastdrive=j command. The default of five drive names is more than enough.

The number of buffers has been reduced from thirty to twenty to conserve memory. I don't need as many buffers because DESQview does its own memory management. The environment is still expanded, but it doesn't need to be expanded as much. In my normal configuration, I use a long Path command; however, DESQview changes the path as needed, so the default path doesn't have to be quite so long.

The one additional device driver, QEXT.SYS, is packaged with DESQview. It creates a small RAM disk in PC AT extended memory to which DESQview swaps out some of its operating code. The extended memory is on one of the memory boards, which has been configured to have 128K of PC AT extended memory, the rest being assigned to expanded memory.

When I need a quick start, I boot the computer from a diskette that has some of my device drivers in its CONFIG.SYS file as well as an abbreviated AUTOEXEC.BAT file. The point of this variation is not so much to save memory as to save time. This CONFIG.SYS file follows. The main expanded memory board is omitted because it takes quite a long time to test and initialize itself. However, the second expanded memory board is included because it holds the nonvolatile RAM disks. Note that although the file appears on a diskette in drive A, the device drivers are loaded from the hard disk. The fraction of a second it takes DOS to find the drivers there is considerably less than the extra time it would take to read them from a diskette.

CONFIG.SYS File for a Quick Start

COMMANDS	COMMENTS
`device=c:\x2cemm.sys`	Driver for second expanded memory board
`device=c:\x2cdrive.sys 0,448,ne;1,96,nre`	Nonvolatile RAM disk driver
`device=c:\msmouse.sys /1`	Mouse driver
`shell=c:\command.com /p /e:1088`	Expands environment to 1088 bytes
`files=20`	Allows up to twenty files to be open
`buffers=30`	Establishes thirty memory buffers
`break=on`	Establishes frequent break-checking

Warnings

You cannot include any extraneous material in a CONFIG.SYS file using a version of DOS earlier than 4.0, although a blank line is acceptable. (In DOS 4.0 you can add remarks to a CONFIG.SYS file to explain the action of a command, by beginning the remark with the Rem command. You can also use the Rem command to disable a command temporarily without deleting it from the CONFIG.SYS file.) With these exceptions, anything other than a legitimate configuration command will result in the message

```
Unrecognized command in CONFIG.SYS
```

However, some unrecognized commands may lock up your computer.

Tips

Buffers, file control blocks (FCBs), and device drivers all take up part of your RAM. If some of your programs are especially memory-hungry, you may need to create a special CONFIG.SYS file for them that requires less memory.

When adding device drivers to your CONFIG.SYS file, be sure that you have a bootable diskette with a copy of your current CONFIG.SYS file, or with no CONFIG.SYS file. Since errors in your CONFIG.SYS file may lock up your computer, you may be forced to do a cold boot. However, you won't be able to start your computer again to fix the CONFIG.SYS file unless you have a diskette that will boot the computer.

For related information, see Break, Buffers=, Country=, Device=, Drivparm=, FCBs=, Files=, Install= LastDrive=, and Shell=.

Copy

Syntax

{**1**} copy *source* [*target*][/v]
{**2**} copy *file1.ext* + *file2.ext* + . . . + *filen.ext*
[*target.ext*][/v][/*type*]
{**3**} copy *source device*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 1.0 to 3.3
{**all**} IBM DOS 4.0

COPY C

Overview

The Copy command copies or combines files. It can also be used to create files and to redirect them (see Copy CON). You can make duplicate copies of files on the same directory and copy files to other drives and directories. If the files are on the same directory, you must give the copy a new name; if they are not, you can either copy files to new files having the same name or to new files with new names. When you do not change names, the source file will replace the file on the target if a file of the same name as the source file exists on the target drive and directory. When you do change names, the source file will replace the file on the target if a file of the same name as the target file exists on the target drive and directory. In neither circumstance are you warned if a file of the same name exists on the target.

When you use the Copy command to combine files, you can either append a group of files to an existing file or combine the files into a new file. Switches allow you to specify whether the copy will include the end-of-file mark (EOF) in a text file or whether the copied material will end at the last character before the EOF.

The Copy command always sets the archive attribute of the new copy of a file (see Attrib and File Attributes). It does not change the attributes of the original.

Procedure/ Examples

Copying Files

{1} `copy source [target][/v]`

To copy files, enter the Copy command followed by the source of the file to be copied and, optionally, the target. The source must include a file name and/or path name and may include a drive name. The target may be a drive name, a path name, a file name, or any combination of these elements.

The Copy command follows a set of rules regarding the name of the source and the target:

1. If no drive or path is specified, the default drive and directory are assumed.
2. If the source file is not on the current drive and directory, you must precede the source file name with the drive and/or path name of its location.
3. If you do not specify a target, the default directory is assumed.

4. If the target is a drive name, files will be copied to the current directory of that drive. Their names will not be changed.
5. If the target is a path name, files will be copied to the specified directory. Their names will not be changed.
6. If the source and target are both the current directory, you must enter a target file name. Otherwise, you will see the message

```
File cannot be copied onto itself
0 File(s) copied
```

You can use wildcard characters to copy a group of files with related names. If you do so, any files on the target with names that match those being copied will be replaced by the new copies.

The /v (Verify) switch checks the integrity of the copy. Including the /v switch is the same as turning on Verify, except that Verify is on only during the execution of its Copy command (see Verify).

Combining Files

{2} copy *file1.ext + file2.ext + . . . + filen.ext*
[*target.ext*] [/v] [/*type*]

To combine files, enter the Copy command followed by the names of the source files to be copied, separated by plus signs. You may omit spaces before and after the plus signs and before the switches.

As with other uses of the Copy command, you may include a drive name and/or directory name as part of the source. If you do not specify a target file name, all the files to be combined will be appended to the first file name in the list.

There are two additional switches for this version of the Copy command. The /a (for ASCII) switch tells DOS to copy the specified files up to the first Ctrl-Z (EOF). This is the default if you do not specify a switch. The /b (for Binary) switch causes the Copy command to treat Ctrl-Z like any other character.

You need to understand the end-of-file mark to understand the importance of these switches. The end-of-file mark is a holdover from CP/M, the operating system on which DOS was originally based. In CP/M, files are saved in 128-byte records. Programs designed for CP/M routinely placed a Ctrl-Z at the end of their data files so that they would know where the file actually ended if its length wasn't an exact multiple of 128 bytes. In

DOS, however, the true length of the file is recorded in the file's directory entry; thus, the end-of-file mark shouldn't be necessary. However, some programs that were originally designed for CP/M (such as WordStar) still routinely place an EOF at the logical end of the file. Programs that do so will not read past the first Ctrl-Z encountered. (Neither will DOS, in many circumstances.) Therefore, if you want to combine several text files generated by such a program, it is essential not to include any extraneous Ctrl-Zs if you want to be able to edit the file.

In contrast to a text file, each byte in a program file represents part of an instruction, not a character. What would be a Ctrl-Z in a text file is merely a byte with the value of 26 in a program file. Therefore, you do not want to treat such bytes as EOFs.

The syntax for these switches is tricky because you must place the switch after the first file name to which you want it to apply. However, the switch also affects all subsequent file names until you enter another switch. Consider, for example, the following command:

```
copy file1.com/b+file2.com+file3.txt/a newfile
```

This command would treat FILE1.COM and FILE2.COM as program files and FILE3.TXT as a text file. The three files would all be combined in the file NEWFILE. If any other files were included in the command after FILE3.TXT, they too would be treated as text files, and the final Ctrl-Z would be removed.

You should be aware of the effect of this type of Copy command on the source files. If you specify a target file, all source files will still be on the disk, as will the new file. If you do not specify a target, the first file named in the series will contain copies of all the files in the command. The old file that had the first file name will be at the beginning of the new file, but all the other source files will still be present.

Combining files with Copy commands that use wildcards is tricky. They follow these rules:

1. Files matching the wildcard pattern will appear in the combined file in the order in which their names appear in the directory. (That order can change each time you write or erase a file.) If this order is not satisfactory, you should list the files by name rather than using a wildcard pattern.
2. If you enter only a wildcard pattern, such as `copy *.txt`, the first file matching the pattern will become the target file.
3. If the name of the target file matches the wildcard pattern, that file will not be included in the combined file. You will see the message `Content`

`of destination lost before copy`, unless it happens to be the first file in the directory that matches the pattern. Thus, if you enter `copy fig4.* fig4.set` and the file FIG4.SET is present in the directory, but is not the first file with the name FIG4, it will be overwritten by the new file. Figure 14 shows what happens with a command of this type. The before-and-after directories show that when the command `copy fig4.* fig4.set` was entered, all the files named FIG4 were copied into FIG4.SET, which no longer exists in any form. The size of FIG4.SET in the second directory display is equal to the sum of the bytes in FIG4.BAT, FIG4.DSP, FIG4.ORG, and FIG4.SCR.

4. If the target file name includes wildcard characters, several files may be created, each of which combines similar files matching the pattern (see Wildcard Characters). However, no files will be copied unless

```
A:\>dir

 Volume in drive A has no label
 Directory of  A:\

NEWDOC   BAK     7808  12-17-87  10:55p
NEWDOC   TXT     8448  12-17-87  10:59p
FIG4     BAT       62  10-13-87  10:15a
FIG4     DSP     1861  10-13-87  10:00a
FIG4     ORG     1867   9-18-87   5:27p
FIG4     SCR     8128   9-18-87   5:31p
FIG4     SET     4256  10-13-87  10:14a
        7 File(s)    335872 bytes free

A:\>copy fig4.* fig4.set
FIG4.BAT
FIG4.DSP
FIG4.ORG
FIG4.SCR
FIG4.SET
Content of destination lost before copy
Content of destination lost before copy
        1 File(s) copied

A:\>dir

 Volume in drive A has no label
 Directory of  A:\

NEWDOC   BAK     8448  12-17-87  10:59p
NEWDOC   TXT     9600  12-17-87  11:15p
FIG4     BAT       62  10-13-87  10:15a
FIG4     DSP     1861  10-13-87  10:00a
FIG4     ORG     1867   9-18-87   5:27p
FIG4     SCR     8128   9-18-87   5:31p
FIG4     SET    11916  12-17-87  11:15p
        7 File(s)    327680 bytes free

A:\>_
```

Figure 14: Combining files using wildcard patterns when the target exists

there is a file matching each wildcard pattern in the list. Suppose, for example, you have a directory containing the files

TEST.OBJ
TEST.MAP
TEST.LST
TEST2.LST
NEWTEST.OBJ
NEWTEST.LST
TEMP.LST
TEMP.MAP

If you enter the command

```
copy *.obj + *.map + *.lst *.all
```

only two new files, TEST.ALL and NEWTEST.ALL, will be created, as Figure 15 illustrates.

Redirecting Files

{3} `copy` *`source device`*

To redirect files to a device rather than to a file, make the target a DOS device name. Thus, to print files from disk, you can enter a command of the form

```
copy filename.ext lpt1:
```

Similarly, to direct files to the second serial port, use a command of the form

```
copy filename.ext com2:
```

The colon is optional in DOS 3 and later.

Warnings

If you copy a file to another directory without changing its name, and the source is too large to fit in the remaining disk space, the file on the target will be erased, even though the source is not copied. Thus, no file by that

```
D:\>copy *.obj + *.map + *.lst *.all
TEST.OBJ
TEST.MAP
TEST.LST
NEWTEST.OBJ
NEWTEST.MAP
NEWTEST.LST
        2 File(s) copied
```

Figure 15: Combining files using wildcard patterns

name will exist on the target. DOS begins copying before it checks available disk space. If it cannot successfully copy a file, it erases its name from the target. Therefore, not only will the source file not be copied, but the original file of the same name on the target will be overwritten.

If the target of a Copy command looks like a path name but there is no path with that name, every file matching the source will be copied in turn to a file matching the name of the target. The target will then be a single file in the directory one level above the presumed target. Thus, if you enter the command `copy *.* c:\reports\temp` and there is no such directory, each file in the current directory will in turn be copied to a file called TEMP in the C:\REPORTS directory, overwriting the previous TEMP. When the process ends, the file TEMP will contain only the last file listed in the current directory.

Tips

To avoid overwriting existing files on the target, use the Exist command in a batch file (see If and Batch Files). For example, to copy a file called NEWFILE.TXT and ensure that a file of the same name is not overwritten, you might use a pair of commands such as these:

```
if exist a:newfile.txt copy newfile.txt a:newfile.bak
if not exist a:newfile.txt copy newfile.txt a:
```

When copying diskettes, you can rearrange the files' clusters so that they appear contiguously on the new diskette by first formatting the target diskette and then entering the command

```
copy *.*
```

In contrast, the DiskCopy command copies diskettes sector by sector, without changing the relative location of any of the data. Files load more quickly when their sectors are contiguous on disk than when they are fragmented.

If you use DOS 3.2 or later, you can use the more efficient Xcopy command to copy groups of files (see Xcopy). Xcopy can copy the matching files in subdirectories of a specified directory as well as in the specified directory and can recreate the directory structure on the target.

Another advantage of Xcopy is that it is much faster than Copy since it reads as many matching files as memory will hold before it begins writing. In contrast, Copy reads one file at a time, and writes it before reading the next.

The gain is most noticeable when using a single physical drive to copy a group of files from one diskette to another. With the Copy command, you must swap diskettes after each file is read and again after it is written. With Xcopy, however, you need only swap diskettes when memory becomes full.

You can use the Copy command to change the date and time stamp on a file in a directory listing by using the syntax

```
copy filename.ext + ,, /b
```

This copies the file to itself, concatenating it with a *null file*. The date and time appearing in the directory will be the date and time when you entered the command.

You can refer to the source as a single period to copy all the files in a directory:

```
copy . a:
```

A single period always represents the current directory (see Directories).

In some versions of DOS, you can copy all the files in a directory by specifying the directory as the source, without using the *.* wildcard pattern.

Copy CON

Syntax

{**1**} copy con *filename.ext*
{**2**} copy con *device*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 1.0 to 3.3
{**all**} IBM DOS 4.0

C COPY CON

Overview You can use a variant of the Copy command to use the console (the keyboard and screen), unaided by any software, as a primitive text editor. You create text on the screen, then direct it either to a file or to a logical device (see Device Names).

Procedure

{1} copy con *filename.ext*
{2} copy con *device*

To create text on the console, type the command copy con, followed by the destination of the text. The destination may be either a file or a logical device.

The cursor will move to the next blank line. Enter one line of text at a time. You can edit the line with the Del and Backspace keys until you press Return. When you press Return, the cursor moves to the next line and you can no longer edit the previous one.

When you reach the end of the text you want to create, press Ctrl-Z, then press Return. (If you have not modified your function keys, you can press F6 instead of Ctrl-Z.) When you press Return, the message 1 File(s) copied will appear.

Examples Copy CON is somewhat simpler than a text editor for creating a short file or testing a device. For example, you might use Copy CON to create a short batch file. NOBAK.BAT, shown in Figure 16, erases backup files with several types of extensions created by different programs. NOBAK.BAT is short enough to create with the Copy CON command.

To ascertain if a device is working, use Copy CON to send a short string to that device. To test the printer, for example, you can enter

```
C:\BATCH>copy con nobak.bat
echo off
del %1 *.bak
del %1 *.~*
del %1 *.$bk
^Z
        1 File(s) copied

C:\BATCH>_
```

Figure 16: Creating NOBAK.BAT

```
copy con prn
This is a test
^Z
```

You know that the printer is properly connected and configured if the text appears on it.

Copy35 and Rcv35

Syntax

{1} copy35 [*source*]*filename.ext* [*target*]
{2} rcv35

DOS Versions

PC-DOS 3.3

Overview

Copy35 and Rcv35 are special commands included in PC-DOS 3.3 that allow you to take advantage of a special feature of the parallel ports on PS/2 computers (see Ports, Serial and Parallel). These parallel ports—unlike all others except for a few on some of the new expansion boards—can receive as well as transmit data; if you have an IBM Data Migration Facility cable, you can use the Copy35 and Rcv35 commands to transfer files from a conventional PC to a PS/2 (or from one PS/2 to another), by way of their parallel ports.

Connect the parallel ports of the two computers using the Data Migration Facility, then enter the Copy35 command on the sending computer and the Rcv35 command on the receiving computer.

Procedure

{1} copy35 [*source*]*filename.ext* [*target*]
{2} rcv35

Copy35 and Rcv35 are external commands; using them requires having the DOS program COPY35.COM on a disk in the sending computer and RCV35.COM on a disk in the receiving computer. If either program is not in its computer's current directory or on its search path, preface its command with a drive name and/or path name; for example,

```
a:copy35 c:\reports\*.rpt
```

or

```
c:\dos\rcv35
```

To transfer files from one computer to another using the Data Migration Facility, hook up the parallel ports of the two computers using the cable. Enter the command rcv35 on the receiving computer, and a command of the form copy35 [*source*]*filename.ext* on the sending computer. You must specify a source that includes a file name, which may contain wildcard characters (see Wildcard Characters). You can also specify a drive and/or path name. If you want to send the files to a particular disk or directory on the receiving computer, specify the target as a drive and/or path name.

Neither Copy35 nor Rcv35 creates directories. To send files to particular directories on the receiving computer, you must create them before beginning the transfer.

Depending on how you enter the source, you can copy entire directories, groups of files in a directory, or single files.

The syntax for Copy35 is the same as for the Copy command, except that you cannot rename files while transferring them.

Examples

To transmit all the files from a single directory, enter a command such as copy35 c:\invoices\qtr3*.* or copy35 c:\reports\qtr1\.

To retain the same directory structure on the target computer, first make sure that the target has equivalent directories, then specify them in the Copy35 command:

```
copy35 c:\invoices\qtr3\*.* c:\invoices\qtr3
```

Warnings

You cannot transfer copy-protected files or files not created under DOS using Copy35 and Rcv35. Neither can you transfer files to RAM disks or to logical drives created by the Assign, Join, or Subst commands, or the DRIVER.SYS device driver.

Do not transfer operating system files and such system-related files as AUTOEXEC.BAT and CONFIG.SYS. They will transfer successfully, but will probably not be appropriate for the receiving computer. If you inadvertently transfer such files, you can restore the receiving computer to its proper configuration using the Replace, Restore, or Sys commands to copy the appropriate files from backup disks.

As with the Copy command, files from the source will replace files on the target if there are files with the same name on both computers.

Do not erase everything from the source computer until you are sure that the files have been successfully transferred.

Tips

You can use the Copy35 and Rcv35 commands to create a different organization of files on the target computer by creating different directories on the target, then selectively copying groups of files to the appropriate new directories. For example, if all your word processing files are in directories called \LETTERS, \MEMOS, and \REPORTS, and you want to reorganize them by topic, you can create directories on the target with names such as \PROJECT1, \PROJECT2, \CHICAGO, and \SALES. Make the various word processing directories current on the source and carefully use wildcard patterns to group all the letters, memos, and reports for each subject in an appropriate directory. Depending on your file-naming conventions, you might use commands such as

```
copy35 c:\letters\*.chi c:\chicago
copy35 c:\reports\??????ch.* c:\chicago
```

and so on (see Directories).

For further information on the syntax and effects of Copy35, see Copy.

Country =

Syntax

{1} country=*nnn*
{2} country=*nnn*[,*charset*]
{3} country=*nnn*,[*charset*],[*drive*][*path*\]country.sys

DOS Versions

{1} MS-DOS 3.2
{1} PC-DOS 3.2
{2}{3} MS-DOS 3.3 to 4.0
{2}{3} PC-DOS 3.3
{2}{3} IBM DOS 4.0

Overview

The Country= command tells the computer to use the date and time formats, currency symbol, decimal separator (period or comma), accented characters, and capitalization conventions used in a particular country. Country= is a configuration command which is placed in the CONFIG.SYS file (see CONFIG.SYS File).

Procedure/ Examples

To choose a country other than the default, enter a command of the form `country=nnn`, where *nnn* is the international telephone prefix for the country, in your CONFIG.SYS file. Table 7 lists the country codes that the Country= command will accept.

If you want your computer to use one of the alternative language character sets as its default character set, include the character set code in the Country= command. For example, to start your computer with defaults for Belgium and use the international character set, enter the command

```
country=032,850
```

If you include a default character set in your Country= command, you needn't use the NLSFunc and ChCP commands unless you want to be able to switch back to the standard character set (see NSLFunc, ChCP). If you do not include a default character set, see the AUTOEXEC.BAT code in the NLSFunc entry for the other steps needed to make your computer speak Belgian.

{2} `country=nnn,[charset],[drive][path\]country.sys`

The Country= command gets its information about national defaults from the COUNTRY.SYS file (see COUNTRY.SYS). If that file is not on the boot directory, you must tell DOS where to find it by specifying its drive and/or directory. You must also include the name of the file, in case you've written a new file of national defaults that DOS can understand, so that DOS knows which file to use. If you haven't written a new national default file, and COUNTRY.SYS is in the \DOS directory on your hard disk, the Country= command to make your computer speak Belgian should be either

```
country=032,850,c:\dos\country.sys
```

if you want the international character set to be the default or

```
country=032,,c:\dos\country.sys
```

TABLE 7 **Country Codes for the Country= Command**

COUNTRY	COUNTRY CODE
Arabic[1]	785
Australia[2]	061
Belgium[2]	031
Canada (English)[1]	001
Canada (French)[1]	002
China (simplified)[3]	086
China (traditional)[3]	088
Denmark[2]	045
Finland[2]	358
France	033
Germany	049
Israel[1]	972
Italy	039
Japan[3]	081
Korea[3]	082
Latin America[1]	003
Netherlands[3]	031
Norway[2]	047
Portugal[1]	351
Spain	034
Sweden[2]	046
Switzerland (French)[2]	041
Switzerland (German)[1]	041
United Kingdom	044
United States	001

[1] *Available only in DOS 3.3 and later.*
[2] *Available only in DOS 3.2 and later.*
[3] *Available only in DOS 4.0.*

if you don't want the international character set to be the default. (You must include the two commas to show that you have left out the character set code.) Alternatively, you can use

```
country=032,437,c:\dos\country.sys
```

since 437 is the code for the default character set. (See Mode for a table of character-set codes.)

Tips

You will usually also want to use a keyboard code to configure your keyboard for the same country (see Keyb and KEYBOARD.SYS). You will probably also want to use the Mode command to install the character set for the country you choose, if it is different from that of the United States (see Mode).

Since DOS computers are usually configured for the country in which they are sold, you can ignore these procedures unless you want to use the national defaults for a country other than your own.

If you create your startup disk using the Select command, you can select a country code and keyboard code when you do so. The Select command will perform most of the other required steps (see Select).

COUNTRY.SYS

Syntax

{**1**} country=*nnn*,[*charset*],[[*drive*][*path*\]country.sys]
{**2**} nlsfunc [*drive*][*path*\]country.sys

DOS Versions

{**all**} MS-DOS 3.3 to 4.0
{**all**} PC-DOS 3.3
{**all**} IBM DOS 4.0

Overview

COUNTRY.SYS is the *country information file* that contains the information DOS uses to conform the computer to various national languages. For each country that has an acceptable country code, COUNTRY.SYS includes the following information:

- Date format
- Time format
- Decimal separator
- Currency symbol
- Capitalization conventions for accented characters
- Proper alphabetical order

It also contains a table for converting accented characters to uppercase characters in file names when they are entered at the keyboard.

Despite its name, COUNTRY.SYS is not a device driver. However, DOS needs to be able to read it to switch from one set of national conventions to another.

For a list of supported countries and their country codes, see Table 7 in the Country= entry.

Procedure

{**1**} `country=`*nnn*`,[`*charset*`],[[`*drive*`][`*path*`\]country.sys]`

You establish a default nationality for your computer by including the Country= command in your CONFIG.SYS file; the command tells DOS to read the necessary information from the tables in COUNTRY.SYS. However, if COUNTRY.SYS is in the boot directory, you need not refer to it in the Country= command. If it is not, you must include its location in your Country= command so that DOS knows where to look for the information it needs.

If you want to use the standards for the United States or English-speaking Canada, you can leave the Country= command out of your CONFIG.SYS file since these are the default nationalities (see Country= for further information.).

{**2**} `nlsfunc [`*drive*`][`*path*`\]country.sys`

If you do not include a Country= command in your CONFIG.SYS file and later decide to use the ChCP command to load alternative national fonts into memory, you can tell DOS where COUNTRY.SYS is in your NLSFunc command (see ChCP).

Some DOS documentation also implies that you can establish a new set of national defaults with this command. You do not get an error message

if you include a country code after the command illustrated above. However, you may be unable to get any of the alternative fonts to display themselves by any means other than using a `device=display.sys` command in the CONFIG.SYS file, and executing the appropriate DOS commands afterward (see NLSFunc).

Examples If you want your computer to use the defaults for Latin America (with the international character set as the default), and COUNTRY.SYS is not in the root directory of the boot disk, use a command such as

```
country=003,850,c:\dos\country.sys
```

in your CONFIG.SYS file. In this command, 003 is the country code for Latin America, and 850 is the code for the international character set.

Tips For further information, see NLSFunc, ChCP, Mode, and DISPLAY.SYS. The ChCP entry shows which commands to include in the CONFIG.SYS file. The NLSFunc entry includes sample AUTOEXEC.BAT code for changing the national defaults, the Mode entry includes a table of character set codes, and the Country = entry includes a table of country codes.

Ctty

Syntax

`ctty` *device*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview Ctty (change teletype) directs DOS to receive its commands from some point other than the console.

Procedure To operate your computer as a remote computer while using another computer or terminal to issue commands, enter the command

```
ctty comn:
```

where *n* is the number of the serial port to which the input device you want to use is connected.

To restore normal use, thereby returning control to your computer's keyboard, enter `ctty con` from the computer attached to the serial port.

To suppress all screen displays, enter the command `ctty nul` in a batch file. This effectively prevents your computer from being controlled from anywhere and redirects the standard input and output to the null device. (See Device Names.) Do not enter this command until you are sure your batch file works.

After you have passed the lines whose display you want to suppress, enter the command `ctty con`.

Examples

Suppose you are connected to another DOS computer via a telecommunications link. If the telecommunications program on the remote DOS computer allows you to enter DOS commands, you can run that computer remotely by first uploading the programs TAKEOVER.BAT and DISMISS.BAT, shown below. Then issue the command that tells the remote telecommunications program to run a DOS command and tell it to run TAKEOVER.

TAKEOVER.BAT

COMMANDS	COMMENTS
`ctty com1:`	Tells the computer to accept control from the serial port
`cd\`	Makes the root directory current
`command`	Loads a secondary command processor (see Command)

DISMISS.BAT

COMMANDS	COMMENTS
`ctty con`	Returns control to the computer
`exit`	Unloads the secondary command processor and returns control to the telecommunications program

You should back up new and changed files daily when using a hard disk (see Hard Disks, Backing Up and Restoring). However, you may not need to back up every file that changes during the course of a day. For

example, some programs record the date you last used the program in a configuration file, which otherwise remains unchanged. Some word processors create a backup file with an extension such as .BAK each time you modify a file. You don't really need to make backup copies of your backup copies.

Some commercial hard-disk backup programs allow you to exclude either certain files or types of files from your daily backup routine. The DOS backup program does not permit this, but you can use DOS commands to accomplish the same thing. The code in the program that follows includes commands to delete some irrelevant files and, using various utility programs, to change the attributes of others. This effectively keeps those files from being backed up.

All these commands issue messages, but since Echo is off, it's not clear which message results from which command. As a result, the messages tend to be confusing and distracting. To avoid this confusion, all output is directed to the null device until the actual backup program is invoked. At this point, control is returned to the computer so that you can see what the backup program is doing. This portion of code appears in the program that follows.

Part of HDBKUP.BAT, Showing Ctty Commands

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`ctty nul`	Suppresses control of computer
`attrib -a \lastbkup`	Clears archive attribute
`fa \~*.* /clear`	Clears all attributes of hidden file so it can be deleted
`del ~*.*`	Deletes formerly hidden file
`attrib -a \timelog\*.bak`	Clears archive attribute
`del \batch\*.bak`	Deletes backup files
`del \batch\*.~*`	Deletes backup files
`attrib -a \util\*.pgm`	Clears archive attribute
`fa \*.fil /clear`	Clears all attributes of hidden file, so it can be copied

continues

COMMANDS	COMMENTS
`attrib -a \util\key\*.bak`	Clears archive attribute
`del e:*.bak`	Deletes backup files
`del e:*.~*`	Deletes backup files
`replace e:*.* c:\edrive /d`	Updates files
`replace e:*.* c:\edrive /a`	Adds new files
`cd \hdbackup`	Makes backup directory current
`ctty con`	Restores control to the computer
`backup c: /s/m/a/1`	Performs the backup

Warnings

You may not be able to control a computer remotely using the Ctty command (even though the same two computers, linked via telecommunications software, can communicate perfectly). However, this technique actually works in some instances.

If you want to enter a `ctty nul` command, put it in a batch file. Before you do so, make sure that a `ctty con` command will be executed at some later point. Otherwise, you will have to turn your computer off and on to regain control.

Date

Syntax

{**1**} `date`
{**2**} `date` *mm/dd/yy*
{**3**} `date` *mm-dd-yy*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 1.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Date command displays the current date as DOS understands it, then allows you to change it. If your computer does not have a

real-time clock, it understands "current date" as January 1, 1980 until you tell it otherwise.

A real-time clock comes with either a program or a device driver that reads the time on that clock and transmits the current date to DOS. If you have a PC AT or compatible, or a 386 PC, the date is stored in the CMOS chip and is automatically transmitted to DOS when you boot the computer.

The current date as DOS knows it appears in the directory for every file that is written. For several reasons, it is important that your computer knows the current date. First, many programs take certain actions based on comparing the directory date of particular files with the current date as known by DOS. Second, programs used to back up a hard disk may need to know the date a file was created or changed when determining which files to back up (see Backup). Third, you may need to know the correct date on which a file was last changed to find a particular iteration of the file.

Procedure {1} `date`

If you do not use an AUTOEXEC.BAT file, your computer displays a prompt of the form

```
Today is mm-dd-yyyy
Enter new date:
```

before displaying the DOS sign-on message (see AUTOEXEC.BAT File). However, if you do use an AUTOEXEC.BAT file, neither the date prompt shown above nor the DOS sign-on message appears and DOS immediately begins executing the commands in the AUTOEXEC file. The date known to DOS then becomes your default system date. This is no problem if you have either a real-time clock or a PC AT or compatible, or a 386 PC. If you do not, include the command `date` in your AUTOEXEC file, preferably near the beginning. This command displays the same date prompt that appears automatically when you do not use an AUTOEXEC file and allows you to enter a new date.

If the date displayed in the prompt is correct, press Return and it will remain the system date.

In DOS 3.0 and later, the date prompt includes sample syntax, such as

```
Today is 6-20-1989
Enter new date (mm-dd-yy):
```

{1} date
{2} date *mm/dd/yy*
{3} date *mm-dd-yy*

To change the system date, enter the Date command at a DOS prompt, either with no parameters or with the desired date, in one of the acceptable forms, as a parameter. If you enter the Date command with no parameters, the current date will be shown, with a prompt to enter a new date. If you enter a date parameter, no messages will appear. You need not include leading zeros in the month and the day.

If you enter the date in an unacceptable format, the prompt

```
Invalid date
Enter new date:
```

appears. In DOS 2, the formats *mm-dd-yyyy* and *mm/dd/yyyy* generate this error message, even though the date display includes a four-digit year; in DOS 3.0 and later, they do not.

The date information is stored in the directory as a two-byte hexadecimal number in bytes 24 and 25 of the directory entry. To decode this number, convert to decimal and use the following formula:

```
DATE = day + (month x 32) + ((year - 1980) x 512)
```

Examples

Sometimes your programs need direct access to the date. The best way to arrange this is to place the date in the environment by using the Set command. The TODAY.BAT program in the Redirection and Pipes entry places these numbers in the environment. You will see a use for these files in the HDBACKUP.BAT program in the Hard Disks, Backing Up and Restoring entry, and in the LOGON.BAT program in the Call entry.

Warnings

Some versions of DOS don't remember to change the date at midnight. If you work past midnight regularly or leave your computer running and

want your directory dates to be accurate, you may have to either reset the date with the Date command or reboot the computer.

Tips

If you have nationalized your computer to a country other than the United States, the format in which you enter the date and the format in which DOS displays it will be that of your selected country.

To change the directory date of a file, use the Date command to set the system date to the date you want the file to have, then open and close the file. Finally, use the Date command to reset the system date to the current date.

In DOS 3.3 and later, the Date command writes directly to the CMOS memory chip on a PC AT or compatible computer, or a 386 PC. You can thus set the date permanently with it, instead of using the Setup program.

For related information, see Time.

DEBUG

Syntax DEBUG is a DOS utility program with many commands of its own. Since each procedure in DEBUG uses different commands, the syntax is listed separately for each operation.

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

DEBUG is a programmer's tool that allows you to examine bytes anywhere in your computer's memory and to change bytes anywhere (except in ROM). It can execute a program one instruction at a time to verify that the program is operating correctly. It can also trace execution, compare values before and after an instruction, and compare and move ranges of data in memory. Finally, it can read and write files and disk sectors.

Many of these functions are useful primarily to assembly-language programmers. However, nonprogrammers can use DEBUG to

- Patch program files
- Examine memory
- Explore the contents of disks
- Alter directory entries (see File Attributes)
- Create short .COM files, either by using DEBUG's Assemble command or by entering script files
- Examine program instructions using the Unassemble command
- Undelete short files (although this is better done with a commercial utility designed for the purpose)
- Recover files that are stranded in memory

The following section provides a nontechnical introduction to DEBUG.

Data in DEBUG DEBUG treats all data as a series of bytes. Thus, it can read any kind of file: program files, text files, and more complex types of data files. DEBUG recognizes two kinds of data—hexadecimal and ASCII. Its display format shows the hexadecimal value of each byte and the ASCII equivalent of bytes that have a value between 32 and 126 decimal.

There are two ways to enter data in DEBUG—prompted and unprompted. With the prompted method, you enter a command that lets you enter data, followed by the address at which you want to enter it. You then see the current contents of the address and a colon prompt. You can enter a new value at the prompt or press Return or Ctrl-C to return to the hyphen prompt.

With the unprompted method, you specify a memory address at which you want to enter data and the bytes to be entered. Unlike in a word processor or text editor, you cannot move the cursor to an entry point and enter or change data; instead, you enter a series of bytes at once or one byte at a time. These entry methods will be explained in detail as appropriate.

DEBUG always expresses addresses as four-place hexadecimal numbers and hexadecimal data as two-place numbers. However, you may omit the leading zeros if the address you are entering is less than four places long or the data only one place.

If you often find yourself mucking about with bytes, you may want to invest in a *byte editor*. A byte editor lets you alter specific bytes by moving a cursor to them and entering the new values. Many commercial utility

collections include a byte editor. It is much easier to make small changes in a program file with a byte editor than with DEBUG.

Memory Addresses You can use DEBUG only with data in memory, and you generally specify the memory address with which you want to work. (Addresses are entered in the SEGMENT:OFFSET notation described in the Memory entry.) If you do not enter an address, DEBUG defaults to the current memory segment, beginning at byte 100H (256 decimal). The first 100H bytes are reserved for the *program segment prefix*, a special area that establishes the linkage between DOS and the program. As a general rule, you can use just the offset address unless you are working with a program that is longer than 64K.

General Rules for DEBUG Commands Once you enter DEBUG, the prompt is a hyphen, and you enter your commands next to this hyphen. Some forms of the DEBUG commands display a single memory location and present a colon as a prompt. At these prompts, you can change the value in the displayed location by entering a new value. The value will not change if you press Ctrl-C or Return.

The DOS editing keys work as they do at the DOS prompt; nothing is actually entered until you press Return (see Command Keys). You can use the previous command as a template and edit it to create a new command before entering it.

All DEBUG commands are entered as a single letter; most may be followed by parameters. You may separate the parameters with spaces or commas.

Most DEBUG commands take an address as their first parameter. You needn't separate the address from the command at all. For example, to use the Dump command to look at data at address 200, you could enter any of the following:

```
d200
d 200
d,200
d, 200
```

If you make an error in a DEBUG command, DEBUG will note the location of the error on the following line; for example,

```
-s100 d 12
        ^Error
```

Procedure **Invoking DEBUG**

{1} debug
{2} debug *filename.ext*

DEBUG is an external command; using it requires having the DOS program DEBUG.COM (or DEBUG.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:debug a:temp.com
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\debug
```

When you invoke DEBUG with no parameters, you can

- Load a file using the Name and Load commands (described later in this entry)
- Load a disk sector using the Load command
- Examine memory areas and CPU registers directly
- Assemble new programs

If you create new programs, you will eventually need to name them if you want to save them.

You can optionally enter the file name as a parameter when you want to examine the contents of a file. If the file is not on the current drive and/or directory, preface it with a drive and/or path name.

You cannot examine an .EXE file directly in DEBUG. You can, however, either give an .EXE file a different extension or copy an .EXE to a new file with a different extension, then examine the new file. See Loading a Program in this entry for more details.

Exiting DEBUG To return from DEBUG to the operating system, enter the Quit command. To do so, type `q` at a hyphen prompt and press Return.

Examining Memory

{1} d
{2} d*start-address*
{3} d*start-address end-address*
{4} d*start-address* l *length*

Use the Dump command to examine memory in DEBUG. Unless you instruct it otherwise, the Dump command displays 80H (128 decimal) bytes of memory at a time, in eight lines of 16 bytes each. As Figure 17 shows, DEBUG displays both hexadecimal and ASCII values in separate portions of the screen.

Notice that the display has three columns. At the left is the address in SEGMENT:OFFSET form. The offset address is that of the byte in the first column in the middle section. The middle section displays 16 bytes on each line. They are set up as two groups of eight values, separated by a hyphen for easy reading. At the right is the ASCII display. Any byte with a value less than 32 or greater than 127 is represented by a period.

If you do not specify an address, the display begins at byte 100H of the current segment of memory. If you specify a starting address, DEBUG displays eight lines beginning at the address you specify. You may specify an offset in the current segment, or both a segment and an offset. Thus, d500 displays 80H (hex) bytes of memory beginning at address 500 in the current segment, and d6000:500 displays 80H bytes at address 500H in segment 6000H.

You can specify a range to display more or fewer bytes. There are two ways to do so: enter either the starting and ending addresses of the range to display or the starting address followed by an l and the length of the

```
C:\>debug
-d
3510:0100  53 51 8B 1E 2C 01 8A 2F-B0 20 E8 54 FE 43 8A 07   SQ....../. .T.C..
3510:0110  3C 1B 75 08 43 8A 07 A2-35 20 EB F1 3C 0D 74 05   <.u.C...5 ..<.t.
3510:0120  E8 56 01 75 E8 E8 A1 02-E8 14 01 E8 22 01 74 04   .V.u........".t.
3510:0130  FE CD 75 D4 80 3E A5 05-00 75 03 E8 D5 E7 C6 06   ..u..>...u......
3510:0140  ED 1E 00 E8 50 E7 3A 06-7C 01 74 11 8A C8 E8 96   ....P.:.|.t.....
3510:0150  E9 75 06 3A 0E 7B 01 75-E5 32 C0 EB 02 0C FF 59   .u.:.{.u.2.....Y
3510:0160  5B C3 53 51 52 C6 06 33-20 FF E8 68 00 74 4F 2E   [.SQR..3 ..h.tO.
3510:0170  80 3E 21 05 00 75 1C FF-36 1E 03 B0 20 E8 E1 FD   .>!..u..6... ...
-_
```

Figure 17: The DEBUG Dump command

range. Thus, to display 256 (200H) bytes beginning at address 500H in the current segment, you could enter either `d500 6ff` or `d500 l 200`. You can omit the spaces in the latter command by entering `d5001200`, but it's harder to read that way.

Once you enter the Dump command, each successive Dump command displays the next 80H bytes.

Loading a Program

{1} `n`*filename.ext* [*parameters*]
`l`[*address*]
{2} `debug` *filename.ext* [*parameters*]

To examine or change a program, you must first load it. You can load a program when you invoke DEBUG by including the file name as a parameter. If you did not include the file name as a parameter or if you want to load a different file, use the Name and Load commands. First, enter the Name command, followed by the file name. (You can include a drive and/or path name if the file is not in the current directory.) Then enter the Load command. To look at a file called NEWFILE.COM, enter

```
nnewfile.com
l
```

The Load command behaves differently with different types of files. If the file is a .COM file, it is loaded at the default starting address (100 hex). The first 100 bytes are reserved for the *header*, or *program segment prefix*, which DEBUG creates to establish communication with DOS. If the file is an .EXE file, DEBUG strips off the header and relocates the code to the address in memory specified in the header. See EXE2BIN for more information about the difference between .COM and .EXE files.

If the file has the extension .HEX, DEBUG assumes it is a special hexadecimal format specific to the Intel microprocessors and loads the file at the address specified in the file. All other files are loaded at the address you specify, or at 100H in the current segment if you do not specify an address.

This distinction affects what you can do with an .EXE file. If you want to debug it by watching its execution, you must load it in this manner and let it relocate itself; otherwise, it won't run. If you want to alter bytes in it, you must first change its extension. DEBUG will then treat it like any other

type of file and load it at the default address or the address you specify. With the code at a specific memory location, you can find the items you want to change. To debug a program that requires parameters, enter the parameters when invoking DEBUG in the Name command, or in a second Name command after the Load command. Thus, to trace the actions of a word processor when a file is open, you could use any of the following forms:

```
debug wp.com textfile.asc

debug
nwp.com texfile.asc
l

debug
nwp.com
l
ntextfile.asc
```

Loading Disk Sectors

laddress drive sector n

You can load disk sectors in DEBUG, regardless of their relationship to files. To load disk sectors, you must specify the following:

- The *address* in memory at which the data should appear.
- The *drive* from which the sector or sectors should be read, specified as a number. Drive A is 0, drive B is 1, and so on. Even if you have no drive B, drive C is 2.
- The *sector* number, using the DOS numbering system, of the first sector to load (see Disk Sectors). Remember to specify this number in hex.
- The number of sectors to read from the disk. This can be any number up to 80H (128 decimal).

To examine the boot record on drive C, enter the command

```
l100 2 0 1
```

and then use the Dump command. This command loads one sector (sector 0) from drive C (that is, 2) at address 100H.

Examining and Changing the Contents of Registers

{1} r
{2} r*register*

To examine the current values in all the registers of the microprocessor, enter the Register command with no parameters. You will see a display similar to the one shown in Figure 18. Each register's two-character name is followed by an equal sign and the value it currently stores.

To change the value in a register, enter the Register command, followed by the name of the register to change; for example,

```
rcx
```

You will see a display such as

```
CX 0000
:_
```

You can then enter a new value at the colon prompt. Press Return or Ctrl-C to keep the current value and return to the hyphen prompt.

Writing a File to Disk

{1} w
{2} n*filename.ext*
rcx
rbx
w

If you have already loaded a program in DEBUG and haven't altered its length, you can write it back to disk simply by using the Write command:

```
w
```

DEBUG will respond with a message of the form

Writing *nnnn* bytes

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=00EE  BP=0000  SI=0000  DI=0000
DS=61D8  ES=61D8  SS=61D8  CS=61D8  IP=0100   NV UP EI PL NZ NA PO NC
61D8:0100 FF76FE        PUSH    [BP-02]                        SS:FFFE=20F
-
```

Figure 18: Register display in DEBUG

To write the file to disk with a new name, use the Name command first:

```
nnewfile.com
w
```

If you have either created a new program or modified an existing one in a way that changes its length, you must tell DEBUG its new length. To do so, use the Register command. The length of a file is stored in the BX and CX registers when you load it. If you have created a new file, the value in these registers will be 0. These registers can each hold a four-place hex number. If the file is less than 64K, you need only the CX register, because one less than 64K is 65,535 decimal, which is FFFF hex. If the file is 64 K or longer, enter the numbers in any places to the left of the first four in the BX register. To enter the length, use the Register command. Thus, if the file were 65,535 (FFFFH) bytes, you would enter

```
rcx
```

You would see something like this:

```
rcx
CX 0000
:_
```

Enter the length in hex next to the colon and press Return. If the file were 65,540 (1000AH) bytes, you would enter the 1 in the BX register, following this procedure, and then enter the A in the CX register. When you have entered the correct length of the file, enter the Write command. The entire process looks like Figure 19. When it is completed, you are returned to the hyphen prompt.

Writing to Disk Sectors

waddress drive sector n

```
-rbx
BX 0000
:1
-rcx
CX 0000
:a
-w
Writing 1000A bytes
-
```

Figure 19: Writing a file from DEBUG

To write disk sectors, you must specify the following items:

- The *address* in memory at which the data to write begins.
- The *drive* to which the sector or sectors should be read, specified as a number. Drive A is 0, drive B is 1, and so on. Even if you have no drive B, drive C is 2.
- The *sector* number, using the DOS numbering system of the first sector to write to (see Disk Sectors). Remember to specify this number in hex.
- The number of sectors (*n*) to write to the disk. This can be any number up to 80H (128 decimal).

To write sectors to disk that you have read and modified, use the same parameters that you used to load the sectors into memory. If your Load command was

```
l100 2 0 1
```

your Write command will be

```
w100 2 0 1
```

Note: The only change you should ever make to the boot record of a disk is the names of the hidden system files and the OEM ID. Change these only to make the disk bootable with an operating system that uses different names for these items.

Note also that the unit for writing to disk is the sector, not a range of data in memory. Therefore, the data will be written in 512-byte units. Even if you want to write data containing only 1 or 2 bytes, 512 bytes will be written.

Searching for Data

{**1**} `s`*start-address end-address data*
{**2**} `s`*start-address* `l` *length data*

To search for data in memory, specify the range of memory to search and the data to search for. To search for values in a file, you should first find out how long the file is. Use the Register command to do so:

```
rcx
```

You will see a display of the form

```
CX nnnn
```

where *nnnn* is the length of the file in hex. You will then search from address 100 to the address in the CX register.

If the file is longer than 64K, you will have to check the BX register as well. This will add up to more than one memory segment. You will then have to use a Dump command to find out the address of the current segment, and specify the end of the range by adding the number in BX to the number of the current segment, and using the number in CX as the offset. Thus, if the BX register contains 0002 and the CX register contains 0FF0, and the current segment is 4000, the end of the range will be 4002:0FF0.

Specify the range either by giving a starting and ending address or by entering a starting address and the length of the range.

You can search for hexadecimal data, ASCII characters, or a combination of both. To search for hexadecimal characters, enter the values to search for on the command line, separated by spaces, after you specify the range.

For example, to search for all the carriage-return/line-feed pairs in a file of 20H bytes, you might enter

```
s100 l 20 d a
```

or

```
s100 119 d a
```

DEBUG responds with a list of all the addresses at which the search data appears. You might see a display such as

```
1B0A:000A
1B0A:0018
-_
```

The address will be that of the first character in the search data. If the list of addresses scrolls off the screen, press Ctrl-P to echo the screen display to the printer, then press F3 to reenter the command.

To search for ASCII characters, enter the characters in single or double quotes. If you want to search for some text ending in a carriage-return/line-feed character, you might enter a command such as

```
s100 l 2ff "This is the end of the file." d a
```

To search for quotation marks, enclose them in quotes. You can use either type of quotation marks, provided they match. Thus, to search for the text

```
"Can computers run themselves?" he asked.
```

You would use a command such as

```
s100 l 4d3 ""Can computers run themselves?" he asked."
```

or

```
s100 l 4d3 '"Can computers run themselves?" he asked.'
```

Entering New Values

{1} *eaddress*
{2} *eaddress data*

There are two ways to enter data in DEBUG. If you specify a single-byte address, DEBUG displays the current value of the byte, followed by a period prompt. At this point you have six choices:

- Enter a new hex value at the prompt and press Return. This will record the new value and return you to the hyphen prompt.
- Enter a new hex value and press the Space bar. This will record the new value and display the value of the next byte, with a period prompt.
- Enter a new hex value and press the hyphen key. This will record the new value and display the value of the previous byte, with a period prompt.
- Press Return. This will return you to the hyphen prompt leaving the value unchanged.
- Press the Space bar. This will leave the value unchanged and display the value of the next byte, with a period prompt.
- Press the hyphen key. This will leave the value unchanged and display the value of the previous byte, with a period prompt.

Because the Space bar takes you to succeeding bytes, you can enter a series of values using this method. Similarly, you can step backward through the file with the hyphen key.

When you patch a program, the patching instructions will generally take the form of a series of keystrokes to enter using this entry method.

Figure 20 shows an example. In this figure, a single line is displayed with the Dump command. The entry address is 100 (hex). The first byte contained the value BB, which was replaced with 0. A press of the Space bar brought byte 101, with the value 81, to the entry line. A second press

of the Space bar brought byte 102 to the entry line, where its value was replaced by BB. Then a press of the Return key brought back the hyphen prompt, where the same line was redisplayed to verify the entries.

The second method of entering data is to specify the data on the command line. The data can be either in hex or in ASCII, following the same rules as for the Search command. For example, to enter the same text you just searched for, you might use a command such as

```
e2b4 22 "Can computers run themselves?" 22 "he asked." d a
```

In this command, 22 is the hex value of the ASCII code for the quotation-mark character; the *d* and *a* are a carriage return and a line feed, respectively.

Filling a Section of Memory

{**1**} *fstart-address end-address data*
{**2**} *fstart-address* l *length data*

The Fill command works the same as the Enter command, except that it enters the specified data repeatedly until all the bytes in the range are filled. The Fill command is most commonly used to fill a range with zeros or spaces to clear out any data that may be in memory before entering new data. You could, however, use Fill to enter other data, as Figure 21 shows.

Comparing Sections of Memory

{**1**} *cstart-address end-address compare-address*
{**2**} *cstart-address* l *length compare-address*

The Compare command notes any differences between two ranges of memory of the same length. Specify one of the ranges to compare, in

```
-d100 10f
61F3:0100  BB 81 00 02 1E 80 00 C6-07 24 BA 82 00 E8 2F 00    .........$..../.
-e100
61F3:0100  BB.0    81.     00.bb
-d100 10f
61F3:0100  00 81 BB 02 1E 80 00 C6-07 24 BA 82 00 E8 2F 00    .........$..../.
-_
```

Figure 20: Entering a byte at a time

either of the two standard forms, and the starting address of the range to compare it with. DEBUG will compare the range beginning at the *compare-address* byte by byte with the base range. If there are no differences, you are returned to the hyphen prompt.

Differences are reported in a four-column display. The first column is the address of the byte in the base range, the second is the value it holds, the third is the value of the associated byte in the compared range, and the fourth is that byte's address. The result might look like Figure 22.

Moving Data in Memory

{1} m*start-address end-address target*
{2} m*start-address* l *length target*

```
-f110 1ff "Let's have a party!" d a
-d100 1ff
61F3:0100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
61F3:0110  4C 65 74 27 73 20 68 61-76 65 20 61 20 70 61 72   Let's have a par
61F3:0120  74 79 21 0D 0A 4C 65 74-27 73 20 68 61 76 65 20   ty!..Let's have
61F3:0130  61 20 70 61 72 74 79 21-0D 0A 4C 65 74 27 73 20   a party!..Let's
61F3:0140  68 61 76 65 20 61 20 70-61 72 74 79 21 0D 0A 4C   have a party!..L
61F3:0150  65 74 27 73 20 68 61 76-65 20 61 20 70 61 72 74   et's have a part
61F3:0160  79 21 0D 0A 4C 65 74 27-73 20 68 61 76 65 20 61   y!..Let's have a
61F3:0170  20 70 61 72 74 79 21 0D-0A 4C 65 74 27 73 20 68    party!..Let's h
61F3:0180  61 76 65 20 61 20 70 61-72 74 79 21 0D 0A 4C 65   ave a party!..Le
61F3:0190  74 27 73 20 68 61 76 65-20 61 20 70 61 72 74 79   t's have a party
61F3:01A0  21 0D 0A 4C 65 74 27 73-20 68 61 76 65 20 61 20   !..Let's have a
61F3:01B0  70 61 72 74 79 21 0D 0A-4C 65 74 27 73 20 68 61   party!..Let's ha
61F3:01C0  76 65 20 61 20 70 61 72-74 79 21 0D 0A 4C 65 74   ve a party!..Let
61F3:01D0  27 73 20 68 61 76 65 20-61 20 70 61 72 74 79 21   's have a party!
61F3:01E0  0D 0A 4C 65 74 27 73 20-68 61 76 65 20 61 20 70   ..Let's have a p
61F3:01F0  61 72 74 79 21 0D 0A 4C-65 74 27 73 20 68 61 76   arty!..Let's hav
-_
```

Figure 21: Filling a section of memory

```
-c110 l 8 1e0
61F3:0110  4C  0D  61F3:01E0
61F3:0111  65  0A  61F3:01E1
61F3:0112  74  4C  61F3:01E2
61F3:0113  27  65  61F3:01E3
61F3:0114  73  74  61F3:01E4
61F3:0115  20  27  61F3:01E5
61F3:0116  68  73  61F3:01E6
61F3:0117  61  20  61F3:01E7
-_
```

Figure 22: Comparing ranges of memory

To move data from one part of memory to another, enter the Move command, followed by the range of addresses of the data to move and the target address for the first byte of the range. For example, to move a block of data from addresses 10B0 through 11C0 to address 100, you would enter either

```
m10b0 11c0 100
```

or

```
m10b0 l 110 100
```

If the target address is within the range of bytes to move, DEBUG will move the data first from the addresses that would be overwritten; therefore, you will not lose any data.

This command is especially useful for recovering files that remain in memory when you find yourself back at a DOS prompt without having saved your file. This may happen if, for example, you have to abort at an `Abort, Retry, Fail?` prompt.

Doing Hexadecimal Arithmetic

`h`*value1 value2*

The Hex command yields the sum and difference of two hexadecimal numbers. Under normal circumstances, you should enter the larger value first. For example, to calculate the sum and difference of 10 and 2, enter

```
h10 2
```

The following line shows the sum and difference:

```
0012  000E
```

This command is useful for calculating the length of a file or a segment of memory, given its starting and ending addresses.

Be careful to enter the larger value first. The system does not recognize negative numbers; instead, it deals only in four-place hexadecimal numbers. Thus, if you enter

```
h2 10
```

the result would be

```
0012  FFF2
```

Add 10 to the value FFF2 to get the correct result.

Creating a Program

`a[`*address*`]`

The Assemble command allows you to create assembly-language programs. If you do not enter an address, 100 in the current segment is assumed. The address will be displayed, followed by a cursor ready for you to enter assembly-language instructions. When you press Return, the next available address appears. When you press Return without entering an instruction, you are returned to the hyphen prompt. To save the instructions as a program file, do the following:

1. Note the address at which you pressed Return.
2. Subtract the starting address from that address, using the Hex command.
3. Enter the result in the CX register with the rcx command.
4. Use the Name command to give the program a name. Use the .COM extension.
5. Write the file using the Write command.

If you use script files such as the one shown in the ErrorLevel entry, you will see that they follow exactly these steps. Each line enters a command or value in response to a DEBUG prompt.

Examining Program Instructions

{**1**} u
{**2**} u*start-address* [*end-address*]
{**3**} u*start-address* l *length*

Use the Unassemble command to examine assembly-language code. Entering the Unassemble command with no parameters displays the instructions in the 20 (hex) bytes beginning with address 100. Typing u at the next hyphen prompt displays the instructions in the next 20 bytes, and so on.

When you use the Unassemble command, DEBUG treats all the bytes in the displayed range as though they were assembly-language instructions. Hence, if you are disassembling a program, you may prefer to enter a range equal to the length of the program. To do so, enter the rcx command to find the length of the file, then enter a command of the form

u100 l *length*

where *length* is the value found in the CX register. Alternatively, you can enter the command in the form

```
u100 end-address
```

where *end-address* is the value found in the CX register minus 1. (Your program starts at 100, not 101.)

Before you start, you might also want to use zeros to fill the range where the program will appear. This will keep you from being distracted by irrelevant instructions. However, all the zero bytes will be interpreted as the instruction

```
ADD     [BX+SI],AL
```

If you disassemble FRIDAY.COM, which appears in the ErrorLevel entry, you will see a display such as the one in Figure 23.

Notice the four columns of data in this figure. The first, of course, is the address. The last two comprise the assembly-language instructions and the addresses, registers, and values they refer to. The intervening column represents the actual machine-level code, displayed as hexadecimal values. You can see, for example, that at address 108, the value being moved into AL is 0, and the second column displays B000. At address 10F, the same instruction—moving the value of 1—is represented in the second column as B001. Thus, B0 is the machine-level representation of the instruction MOV AL, and the last two digits represent the value being moved.

```
-n\testprog\friday.com
-l
-rcx
CX 0019
:
-u100 118
61F4:0100 B42A          MOV     AH,2A
61F4:0102 CD21          INT     21
61F4:0104 3C05          CMP     AL,05
61F4:0106 750B          JNZ     0113
61F4:0108 B000          MOV     AL,00
61F4:010A 80FA15        CMP     DL,15
61F4:010D 7306          JNB     0115
61F4:010F B001          MOV     AL,01
61F4:0111 EB02          JMP     0115
61F4:0113 B002          MOV     AL,02
61F4:0115 B44C          MOV     AH,4C
61F4:0117 CD21          INT     21
-_
```

Figure 23: Disassembly of FRIDAY.COM

Program-Debugging Commands The following commands are all advanced commands used in creating, testing, and debugging assembly-language programs. Because of their nature, only an overview is provided here:

COMMAND	COMMENTS
Input	The Input command displays the current value in one of the ports on one of the processor chips in your computer. The addresses of the various ports and their functions vary greatly from one computer to another (even computers of the same brand). To read the value in a port, enter a command of the form `i`*`port-address`* Output
Output	The Output command places a value in a processor port. Enter the address of the port, followed by the value of the byte to be placed there: `o`*`port-address`* `value`
Go	The Go command executes the program in memory, beginning at the instruction pointer in the current segment. It runs as though you had invoked it at a DOS prompt, then continues until it reaches a break point. You can set both the address at which to start executing and up to ten break points by using a command of the form `g=`*`start-address bp bp bp . . .`* where *bp* is the address of the break point. When execution reaches a break point, the next instruction is displayed, along with the values of all flags and registers resulting from the previous instruction.
Trace	The Trace command executes the program producing a display similar to the one produced by Go after each instruction. You can specify a starting address and a number of instructions to perform with a command of the form `t=`*`start-address steps`* The display scrolls continuously until the end of the program is reached or the specified number of steps is executed. To halt the display, press Ctrl-Num Lock or Ctrl-S. To restart it, press any other key except Ctrl-C.
Proceed	The Proceed command executes a specified number of instructions, beginning at a specified address. After the specified number of instructions, it produces the same display as the Go command. Enter a command of the form `p=`*`start-address steps`* Unlike the Trace command, the Proceed command treats a subroutine as a single instruction and produces its display after executing the subroutine. This is useful for finding out whether a subroutine produces the desired result.

These advanced commands are primarily of use to serious programmers. Consult your DOS manual for further details.

Extended-Memory Commands The following commands are all new in DOS 4.0. They are advanced commands used for dealing with the relationship between extended memory and areas of conventional memory. For more details about the matters to which these commands refer see Memory and XMA2EMS.SYS.

EMS Allocate — Allocates the specified number of 16K EMS memory pages and assigns them to a *handle*—a four-digit hexadecimal number that the EMS memory driver uses to refer to a group of pages.

```
xapages
```

EMS Deallocate — Deallocates the EMS pages identified with the specified handle.

```
xfnnnn
```

EMS Map — Maps the memory in an EMS logical page to an EMS physical page. *logicalpage* is one of the 16K pages associated with the handle specified by *nnnn*. *physicalpage* is the address of a specific 16K physical page of EMS memory.

```
xm logicalpage physicalpage hnnnn
```

EMS Status — Displays a report on the number of pages allocated to each handle; the frame segment mapped to each physical page; the total number of EMS pages in the computer, and the number allocated; the total number of EMS handles supported, and the number allocated.

```
xs
```

Examples Here is an exercise using several DEBUG commands. In this exercise you're going to fill a section of memory with zeros, load a disk's boot record into memory, display the boot record, and write it to a file (see Boot Record).

First, invoke DEBUG, without parameters:

```
debug
```

The boot record is a single sector of a disk, and a sector has 512 (200 hex) bytes. Therefore, fill 200 bytes with zeros to ensure a clear work space:

```
-f100 1ff 0
-_
```

Next, you must read a boot record. In this case, we'll read the boot record from the hard disk with the Load command:

```
-l100 2 0 1
-_
```

This command loads one sector, sector 0 from drive 2 (that is, drive C) at address 100.

To examine the result, use the Dump command four times. The procedure so far would look like Figure 24.

To write the boot record to a file, first name the file as follows:

```
-nbootrec.deb
-_
```

```
-f100 2ff 0
-l100 2 0 1
-d
61D8:0100  EB 28 90 49 42 4D 20 20-33 2E 33 00 02 04 01 00   .(.IBM  3.3.....
61D8:0110  02 00 02 FB F2 F8 3D 00-11 00 05 00 11 00 00 00   ......=.........
61D8:0120  00 00 00 00 00 00 00 00-00 00 FA 33 ED B8 C0 07   ...........3....
61D8:0130  8E D8 C4 1E 1C 00 88 16-FD 01 0A D2 79 08 89 1E   ............y...
61D8:0140  24 00 8C 06 26 00 8E C5-8E D5 BC 00 7C FC 1E 36   $...&.......|..6
61D8:0150  C5 36 78 00 BF 2A 7C B9-0B 00 F3 A4 1F C6 06 2E   .6x..*|.........
61D8:0160  00 0F BF 78 00 B8 2A 7C-AB 91 AB FB 8A 16 FD 01   ...x..*|........
61D8:0170  CD 13 A0 10 00 98 F7 26-16 00 03 06 0E 00 E8 73   .......&.......s
-d
61D8:0180  00 E8 79 00 BB 00 05 53-E8 A0 00 5F BE 71 01 B9   ..y....S..._.q..
61D8:0190  0B 00 90 F3 A6 75 57 83-C7 15 B1 0B 90 90 F3 A6   .....uW.........
61D8:01A0  75 4C 26 8B 47 1C 99 8B-0E 0B 00 03 C1 48 F7 F1   uL&.G........H..
61D8:01B0  3D 14 00 7F 02 B0 14 96-A1 11 00 B1 04 D3 E8 E8   =...............
61D8:01C0  32 00 FF 36 24 00 C4 1E-6D 01 E8 30 00 E8 5B 00   2..6$...■..0..[.
61D8:01D0  2B F0 76 0D E8 1D 00 52-F7 26 0B 00 03 D8 5A EB   +.v....R.&....Z.
61D8:01E0  E9 5B 8A 2E 15 00 8A 16-FD 01 FF 2E 6D 01 BE 8B   .[..........■...
61D8:01F0  01 EB 54 90 01 06 24 00-11 2E 26 00 C3 A1 18 00   ..T...$...&.....
-d
61D8:0200  F6 26 1A 00 91 A1 24 00-8B 16 26 00 F7 F1 92 8B   .&....$...&.....
61D8:0210  0E 18 00 F6 F1 2A CC 91-FE C5 86 E9 D0 CE D0 CE   .....*..........
61D8:0220  0A F1 86 F2 87 CA 8A 16-FD 01 C3 BF 05 00 B8 01   ................
61D8:0230  02 CD 13 72 03 B0 01 C3-80 FC 11 74 F8 33 C0 CD   ...r.......t.3..
61D8:0240  13 4F 75 EA BE D5 01 E8-1D 00 BE AB 01 E8 17 00   .Ou.............
61D8:0250  33 C0 CD 16 36 C7 06 72-04 34 12 EA 00 00 FF FF   3...6..r.4......
61D8:0260  B4 0E BB 07 00 CD 10 AC-3C 24 75 F4 C3 00 00 70   ........<$u....p
61D8:0270  00 49 42 4D 42 49 4F 20-20 43 4F 4D 49 42 4D 44   .IBMBIO  COMIBMD
-d
61D8:0280  4F 53 20 20 43 4F 4D 00-00 00 00 0A 0D 4E 6F 6E   OS  COM......Non
61D8:0290  2D 53 79 73 74 65 6D 20-64 69 73 6B 20 6F 72 20   -System disk or
61D8:02A0  64 69 73 6B 20 65 72 72-6F 72 24 0A 0D 52 65 70   disk error$..Rep
61D8:02B0  6C 61 63 65 20 61 6E 64-20 73 74 72 69 6B 65 20   lace and strike
61D8:02C0  61 6E 79 20 6B 65 79 20-77 68 65 6E 20 72 65 61   any key when rea
61D8:02D0  64 79 0A 0D 24 0A 0D 44-69 73 6B 20 62 6F 6F 74   dy..$..Disk boot
61D8:02E0  20 66 61 69 6C 75 72 65-24 43 6F 70 72 2E 20 43    failure$Copr. C
61D8:02F0  4F 4D 50 41 51 20 31 39-38 33 2D 38 37 80 55 AA   OMPAQ 1983-87.U.
```

Figure 24: Loading a boot record in DEBUG

Then tell DEBUG how long the file is. The file is 200 hex bytes, so you must enter this value in the CX register:

```
-rcx
CX 0000
:200
-_
```

Now write the file and quit:

```
-w
Writing 0200 bytes
-q
D:\>_
```

You might think that the file BOOTREC.DEB contains the display shown in Figure 24, but in reality it contains the actual contents of the boot record. If you use the Type command to display the file, you will see something similar to Figure 25. The display ends at the first byte with the value of 1A because the Type command treats this value as an end-of-file mark.

To reproduce the display we saw in DEBUG, load the file in DEBUG and dump it again. Some utility programs display files more flexibly than does the Type command. Figure 26 shows the file displayed in the shareware program LIST.COM, a program all serious DOS users should have. Each byte of the file is represented here by its graphics equivalent.

Several steps are required to recover from memory a file that has not been written to disk. First, use the Dump command to step through memory until you see the file; note the starting address. Continue dumping until you see the end of the file; note the ending address. Use the Hex command to find out how long the file is, then use the Move command to

```
D:\>type bootrec.deb
ð(éIBM  3.3 ◙♦☺ ◙ ◙◙ⁿ²ⁿ= ◄ ♣ ◄                    ·3ФЧ ╚Ä╪─▲╚ ê▄²☺

                                                                  πë▲$ î▲& Ä†Ä╒ ¦ⁿu▲9†6x
ı*¦╢ð ⁽ñ▼╞▲. ╬ı x ╕*¦½æ½Ûè▄²☺═!!á► ÿ~&▄ ♥▲♫ ō5 ōy ╕ ♣SǒáL _ªqÉ╢ð é⁽ªuЩâ╢ß▒▒ðÉÉ⁽ªuL&ï
Ǫ↔ R~&ð ♥╪Zðθ[è.§ è▄²☺ .m☺╓ÏÉðTÉ☺▲$ ◄.& ├í↑ ÷&v
D:\>_
```

Figure 25: Displaying the boot record with the Type command

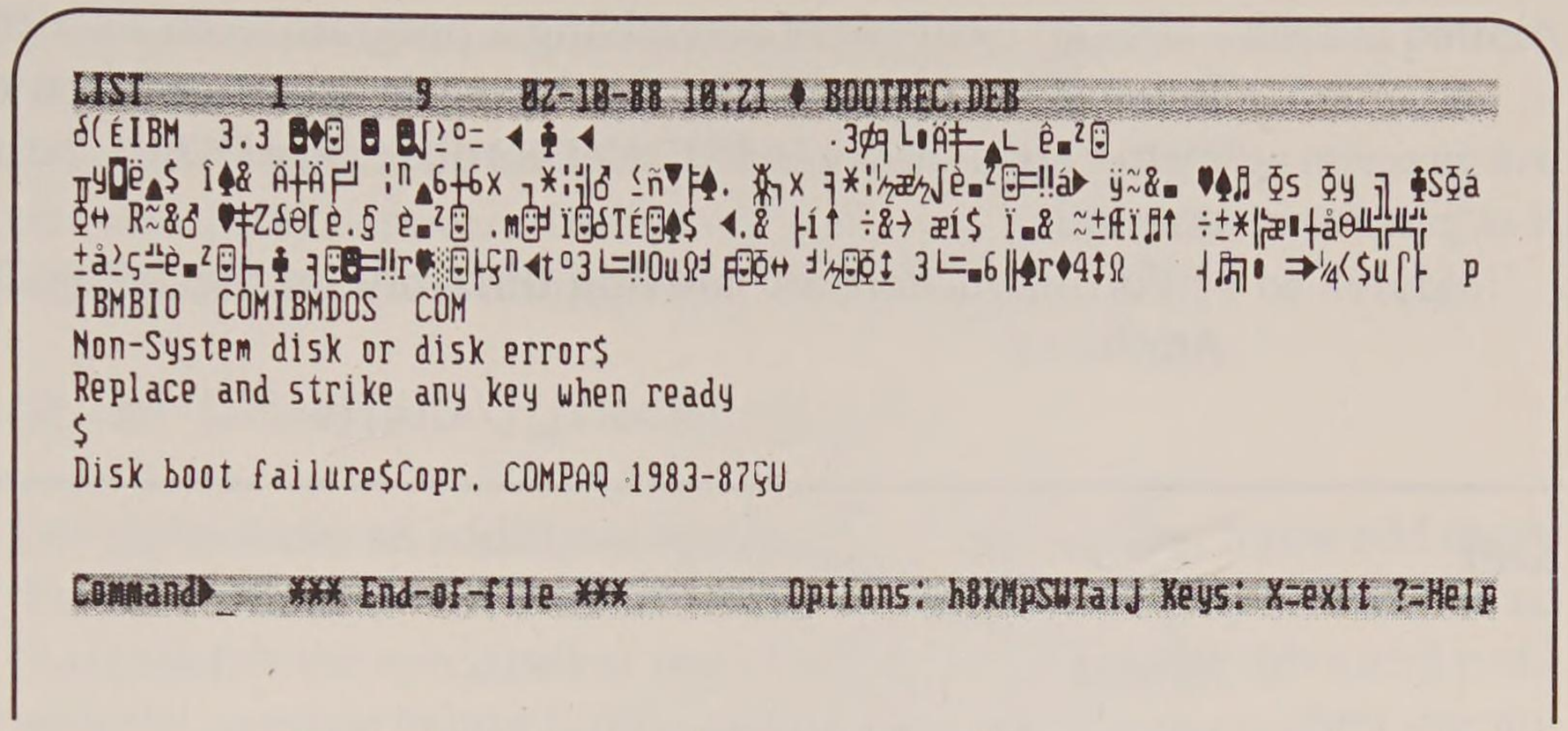


Figure 26: Displaying the boot record with LIST.COM

move the file to address 100 in the current segment. Next, enter the length in the BX and CX registers using the Register command. Finally, name and write the file. To avoid overwriting an earlier version, give the file a name that's not already in use.

Warnings

To make alterations to a program file, copy the program file to a temporary file, then make the alterations to the copy. When you finish, try executing the copy and be sure it works before you delete the original.

When using script files to create programs, be sure to include the Quit command and the final carriage return. When you use redirected input, your computer accepts no input from the keyboard (see Redirection and Pipes). Thus, without the q command in the script file, you are locked into DEBUG until you reboot the computer.

Tips

When DEBUG displays data faster than you can read it, you can halt the display temporarily by pressing Ctrl-S or Ctrl-Num Lock. Alternatively, you can echo the screen to the printer by pressing Ctrl-P or Ctrl-PrtSc.

DEBUG may seem confusing if you're unaccustomed to working with hexadecimal numbers. In the long run, however, it's easier to learn to think in hexadecimal units than to remember decimal equivalents and to convert them accurately.

For an example of assembling a program from a script file, see Error-Level. If you want to try the Assemble command, you could enter the instructions shown in FRIDAY.DEB directly in DEBUG rather than using a script file.

For instructions on altering directory entries using DEBUG, see File Attributes.

Del

Syntax

{1} `del [drive][path\]filename.ext`
{2} `del [drive]path`
{3} `del [drive][path\][filename.ext] /p`

DOS Versions

{1}{2} MS-DOS 2.1 to 3.31
{1} PC-DOS 1.0 to 3.3
{2} PC-DOS 2.0 to 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview The Del (delete) command erases one or more files. Erase is an alias for Del.

Procedure {1} `del [drive][path\]filename.ext`

To delete a file, enter the Del command, followed by the name of the file to be deleted. If the file is not in the current directory, you may precede the file name with a drive and/or path name.

To delete several files with similar names, use wildcard characters in the file name (see Wildcard Characters).

{2} `del [drive]path`

To delete all the files in a directory, you need only specify the path to the directory in the Del command. If the directory is the current directory, you can represent it with a single period.

You can also delete all files in a directory by using the wildcard pattern *.*.

When you ask DOS to delete all the files in a directory, the message `Are you sure (Y/N)?` appears. In DOS 4.0, this message is more emphatic, as it is preceded by the warning `All files in directory will be deleted!`

{3} `del [`*drive*`][`*path*`\][`*filename.ext*`] /p`

DOS 4.0 includes an additional option when deleting files. If you add the /p (for permission) switch to a Del command, DOS will display each file name that matches the specification you entered, complete with drive and path, with the question `Delete (Y/N)?` Using the /p switch is an excellent precaution if you use a command that might erase all the files in a directory.

If the file you specify does not exist in the designated directory, DOS tells you `File not found`. Otherwise, no messages appear.

Warnings

Since the Del command does not tell you which files you are deleting, you should use the command carefully. In particular, be sure that a wildcard pattern does not include the names of files you intend to keep.

If your version of DOS doesn't include the /p switch, to be absolutely sure of the results of a Del command, first issue a Dir command with the same wildcard pattern you intend to use. Then issue the Del command and look at the directory again. Del displays no messages unless it finds no matching files to delete. Therefore, if you see no message you'll know that something was erased. Be sure that what was erased was what you intended and not something else as well.

Device=

Syntax

`device=`*devname*`.sys`

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Device= command is a configuration command used in the CONFIG.SYS file to load device drivers (see CONFIG.SYS File). The following device drivers come with DOS:

VDISK.SYS (in PC-DOS 3.0 and later as well as some versions of MS-DOS)
RAMDRIVE.SYS (in most versions of MS-DOS 3.0 and later)
ANSI.SYS
DRIVER.SYS (in DOS 3.2 and later)
DISPLAY.SYS (in DOS 3.3 and later)
PRINTER.SYS (in DOS 3.3 and later)
XMAEM.SYS (in DOS 4.0)
XMA2EMS.SYS (in DOS 4.0)

Hardware such as external disk drives, expanded memory boards, mice, and real-time clocks, may require device drivers. In addition, some software requires you to install a device driver with the Device= command. Most device driver files have a .SYS extension, although some have a .COM or .BIN extension.

Procedure

Generally, Device= commands should precede other commands in the CONFIG.SYS file. For example, Device= commands that load drivers for physical disks should precede other drivers, and Device= commands that initialize memory boards must precede Device= commands that install devices (such as RAM disks or print spoolers) in the memory on those boards. Each command must be entered flush with the left margin, on a line of its own; for example,

```
device=ansi.sys
device=emm.sys
device=vdisk.sys 512
```

Tips

For a full discussion of device drivers and configuration commands, see CONFIG.SYS File. For details on drivers that are installed with the Device= command, see ANSI.SYS, DISPLAY.SYS, DRIVER.SYS, PRINTER-.SYS, RAMDRIVE.SYS, VDISK.SYS, XMAEM.SYS, and XMA2EMS.SYS.

Device Names

Overview

From the point of view of your computer's microprocessor—its brains, if you will—everything else in the system is a peripheral device. Some peripheral devices have names known to DOS. These names are "reserved" and cannot be used as file names. The reserved device names appear in Table 8. However, DOS can direct output to these peripheral devices. (See Redirection and Pipes.) You can also direct your computer to receive input from some of these devices with the Ctty command (see Ctty). In DOS 2, device names must be followed by a colon. In DOS 3.0 and later, the colon is not needed.

TABLE 8 **DOS Device Names**

AUX	The first serial port, also known as COM1.
COM*n*	A serial, or communications port. Most MS-DOS computers can have up to two. DOS releases 3.3 and later can address four. (*n* represents the logical number of the port.) PS/2 computers can have up to eight.
CON	The console (the keyboard and screen).
LPT*n*	A parallel port. DOS can address up to three parallel ports. *n* represents the number of the port. Because printers are usually attached to a parallel port, the name of this port stands for line printer. Most MS-DOS computers can only output to parallel ports. However, PS/2 computers can also receive input through them. See Copy35 for further information.
NUL	An empty device, also known as the "bit bucket." Anything directed to the Null device will disappear forever.
PRN	An alternative name for LPT1.
PRT	An alternative name for LPT1 in some manufacturers' versions of MS-DOS.
USER	An alternative name for COM1 in some manufacturers' versions of MS-DOS.

Dir

Syntax

{1} dir [*drive*][*path*\][*filename*][*.ext*]
{2} dir [*drive*][*path*\][*filename*][*.ext*] /w
{3} dir [*drive*][*path*\][*filename*][*.ext*] /p

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**1**}{**2**} PC-DOS 1.0 to 1.1
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Dir command displays the contents of a disk directory (see Directories). You can display all or part of a directory in either of two formats, optionally pausing when the screen is full.

Procedure/ Examples

Displaying Complete Directories To display the names of all files in the current directory, enter `dir` with no parameters. This command produces a display that shows the following for each file:

- The file name
- The file name extension
- The size of the file in bytes
- The date it was created or last changed
- The time it was created or last changed

In addition, the directory display includes:

- The volume label of the disk or a message that the disk has no volume label
- The volume serial number, if you are using DOS 4.0 and the disk was formatted with DOS 4.0
- The name of the directory, if it is a subdirectory
- The number of bytes of storage space available on the disk

An example appears in the Backup entry.

To display only the file names and their extensions in wide format (five file names per screen line), use the /w (wide) switch. This switch is useful if your directory has too many files to list on a single screen or if you need only to see if a particular file is in the directory and don't need to knwo when it was modified or how large it is. Figure 27 shows the directory illustrated in the Backup entry in wide format.

Note that subdirectories (including the period entries for the current and parent directories) are also listed only by name. There is no indication that these names represent directories and not files.

To display a directory in standard format, but pausing when the screen is full, use the /p (pages) switch.

To display a directory other than the current directory, include a drive and/or path name in the command; for example,

```
dir d:\report
```

Displaying Partial Directories You can use wildcard characters to display a selected group of files within a directory (see Wildcard Characters). Unlike most commands that use wildcards, Dir acts on all files that match the file name, regardless of the extension. Thus, `dir a*` displays all file names that begin with the letter A, regardless of their extension. Given such a wildcard pattern, other commands act only on file names that begin with A and have no extension.

To display all file names with a given extension, you need not include the file name in the command. The command `dir .com` displays all files with a .COM extension, just as `dir *.com` does.

Tips

To create a handy diskette label, set your printer to condensed mode and enter the command `dir>prn`. This directs the output of the Dir command to the printer. (See Redirection and Pipes.)

For related information see Append, Directories, Extensions, and File Names.

```
 Volume in drive C is JK'sTurboAT
 Directory of  C:\UTIL\PFMNCE

                                MEMSPEED COM     FRP       EXE     FAST      DOG
.               ..
DOG       EXE     BACKSTAT EXE     RAMTEST  COM     SPEED     COM     CLUSTER   COM
RAMTEST   ECT     RAMTEST  OVL     FULL     DOG
        13 File(s)   9125888 bytes free
```

Figure 27: A directory displayed in wide format

Directories

Overview

Directories, along with the file allocation table, enable DOS to keep track of files on disks.

Information in a Directory Entry Directories contain a variety of information about files, and you see some of this information when you use the Dir command. This command typically reveals the following for each file in the directory:

- The file name
- The file name extension
- The size of the file in bytes
- The date the file was created or most recently modified
- The time, to the nearest minute, that the file was created or most recently modified

In addition, in a coded form not revealed by the Dir command, each entry also includes:

- The number of the cluster on the disk where the file begins (see Disk Clusters)
- Information about the file's attributes (see File Attributes)
- The time (to the nearest two seconds) the file was created or most recently changed

Whenever you issue a command to open, copy, rename, delete, execute, or view a file, DOS has to locate the file. It does so by using the starting cluster number along with information in the file allocation table. The file's attributes are used to organize the disk and to keep track of which files have been backed up.

Tree-Structured Directories The main directory on any disk is the *root directory*. It appears in the disk sectors immediately following the file allocation tables.

Root directories can hold only a limited number of file names; the exact limit varies with the size of the disk. (See Diskettes, Structure of, and Hard Disks, Structure of.) However, hard disks and high-capacity diskettes have room for hundreds of files. Therefore, DOS versions since 2.0 allow you to create additional directories (called *subdirectories*) on disks.

Subdirectories may be merely a convenience on high-density diskettes, but they are essential for coherent organization of a hard disk.

Subdirectories allow you to group your files logically. As a rule, you should create a new directory for every software package. (See MD.) You might also want separate directories for different types of files, for files relating to different projects, or for various users. The organization of your subdirectories should depend on the way you work and the software you use.

You refer to subdirectories by using path names, which tell DOS where to find files that are not on the current directory (see Path Names). See CD for details on moving from one directory to another.

When you use subdirectories, your disk is described as having a *tree structure*. The name reflects the appearance of a diagram of a set of directories, which resembles the branches of a tree. (See Figures 28 and 30 for examples.) The root, of course, is the starting point. However, a tree diagram of a disk usually shows the root at the left or at the top rather than at the bottom.

Another analogy used to explain subdirectories is the paper filing system. If you apply this analogy to the diagram in Figure 28, \REPORTS and \INVOICES are file drawers, the regional directories are the hanging folders in each drawer, and the quarterly directories are the manila folders in each hanging folder. However, as we'll see shortly, this analogy can cause you some grief if you take it too literally.

Parents and Children Directories subordinate to a given directory are conventionally called *children* of that directory. Conversely, a directory one level above a given directory is its *parent*. Thus, in Figure 28, REPORTS is a child of the root directory, and EAST, MIDWEST, SOUTH, and WEST are all children of REPORTS. The root directory is the parent of REPORTS, INVOICES, WP, DBASE, LOTUS, and TELECOM.

In every directory other than the root directory, the first two entries take the form

```
.           <DIR>       2-25-88   11:07a
..          <DIR>       2-25-88   11:07a
```

The first of these entries, the single period, refers to the current directory. The double period refers to the parent of the current directory.

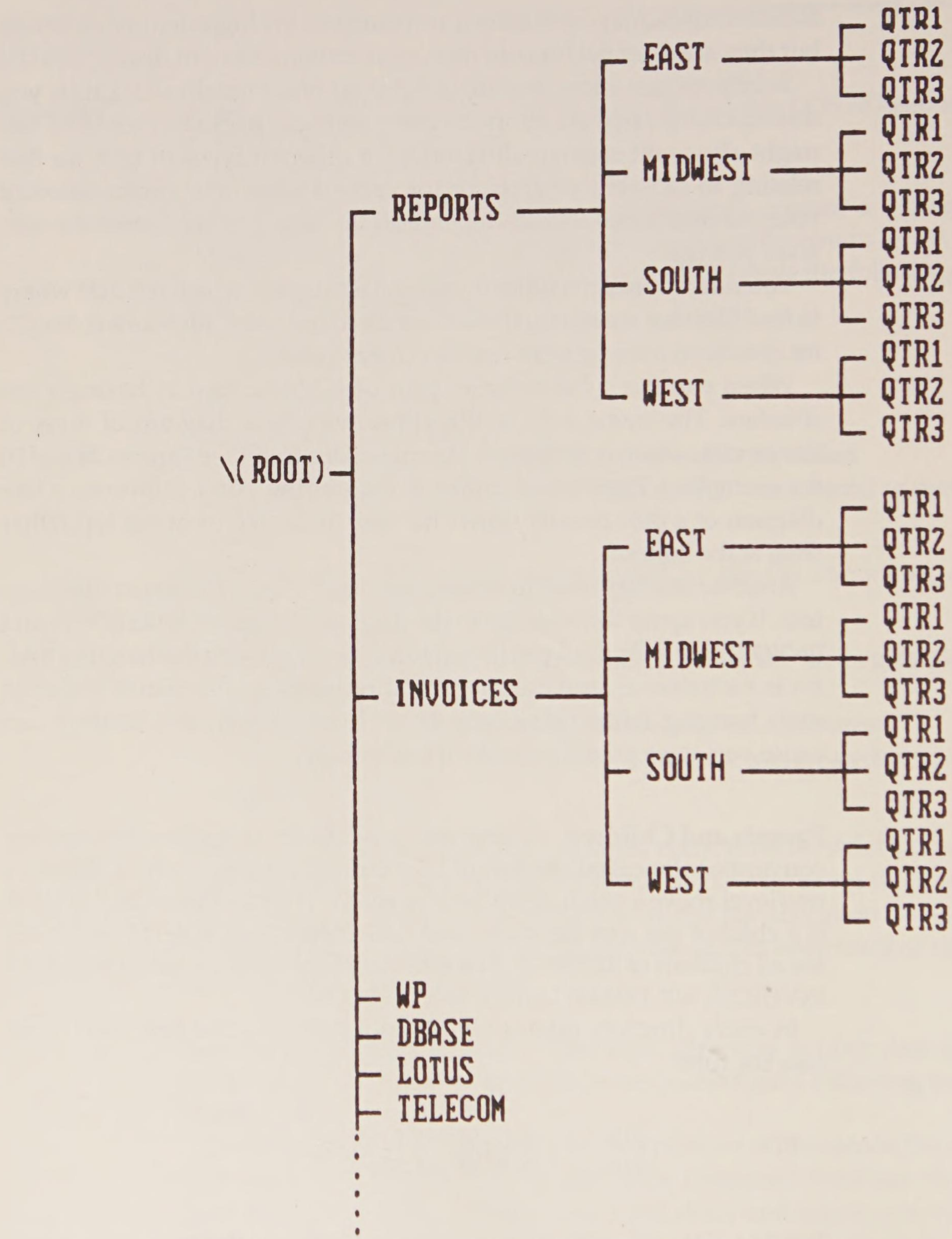


Figure 28: Sample directory tree

The Contents of a Directory Entry on Disk A directory is not a simple ASCII file. Each directory entry comprises a complex set of binary-coded information taking up only 32 bytes. Before looking at all the items in an entry, let's see how DOS stores a directory entry on disk.

Figure 29 shows the structure of a single directory entry, each cell of which represents a byte. For reference, the hexadecimal number of each cell appears in its upper right corner, and the decimal number in the upper left corner, in white on black. In this discussion, bytes are referred to by their decimal numbers.

The file name and extension occupy the first 11 bytes, from byte 0 through byte 10. (The file name appears entirely in uppercase characters since DOS automatically converts all file names to uppercase.)

No separation appears between the file name and its extension; DOS supplies the space that appears when a directory entry appears on the screen. Since you can't type spaces as part of a file name, type a period to show DOS where the name ends and the extension begins. If a file name is less than eight characters, space characters (ASCII 32) fill the remaining bytes.

Byte 11 is the *attribute byte*. This single byte tells whether a given file is:

- A normal file (having none of the other attributes)
- A read-only file
- A hidden file
- A system file
- A subdirectory name
- A disk volume label

Byte 11 also indicates whether the file has been backed up since it was last changed. (See File Attributes.)

Zeros occupy bytes 12 through 21, which constitute the *reserved area*. Nobody seems to know what the area is reserved for, but you risk destroying your directory if you put anything else in these bytes.

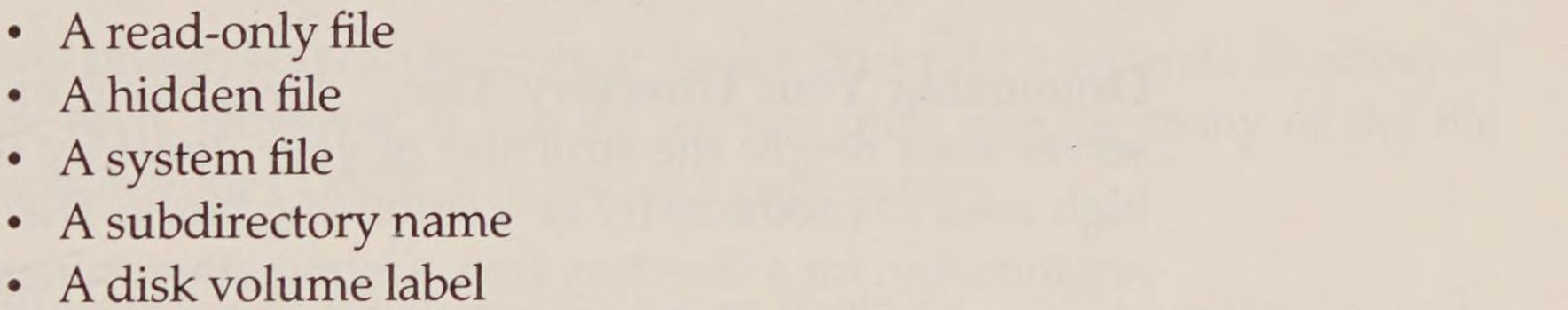


Figure 29: The structure of a directory entry

Bytes 22 and 23 store the time the file was last changed, and Bytes 24 and 25 record the date the file was last changed. This information is represented by specific bits in these four bytes, as explained in the Date and Time entries. When you type `dir`, DOS translates these bits into something understandable. (See Date and Time.)

Bytes 26 and 27 store the location of the first cluster of the file. When you call a file, DOS first looks at these two bytes, then it looks at the byte in the file allocation table representing the first cluster to find the location of the next cluster (see File Allocation Table). DOS locates all the other clusters in the file by reading the file allocation table.

The number of the first cluster is stored in typical Intel low-byte/high-byte form. Thus, to find the actual cluster number, multiply the hexadecimal number in the second byte (byte 27) by 100 hex, then add it to the hexadecimal number in the first byte (byte 26).

Bytes 28 to 31 store the size of the file in bytes. These four bytes follow the low-order/high-order format *in extremis*. The total file size is equal to:

```
byte 28 + (byte 29 x 100H) + (byte 30 x 1000H) + (byte 31 x 10,000H)
```

Procedure

The following steps take you through setting up an effective directory structure that is easy to use.

Organizing Your Directory Tree Depending on whether or not it serves your needs, the structure of your directory tree can be either a high road to productivity or a stumbling block. There is no single ideal organization for a directory tree. Among other things, the best directory structure for you depends upon:

- Your work habits
- Whether or not you have exclusive use of your computer
- The characteristics of the software you use

Regardless of these considerations, however, these principles should guide you:

1. Plan your system in advance.
2. Keep your root directory simple.
3. Maintain a separate entry for each software package that contains multiple files.

4. If your software permits it, keep your data files in directories that bear a logical relationship to your work.
5. Create separate directories for the DOS external programs, for batch files, and for your utility software (see Batch Files). Place all of these directories near the beginning of the search path (see Path).
6. Give your subdirectories names that will still mean something to you six months later.

If you plan your system in advance, you'll have fewer problems later. When planning, remember that DOS does not permit you to rename subdirectories; consequently, you should choose subdirectory names carefully (see Naming Subdirectories, later in this entry).

Maintaining a Simple Structure Your root directory should contain only what your system needs to get itself running as well as its principal subdirectories. Thus, be sure your root directory contains the following:

- COMMAND.COM
- Your AUTOEXEC.BAT file
- Your CONFIG.SYS file
- The device drivers loaded by the CONFIG.SYS file

If you prefer, you can keep your device drivers in a separate directory of their own. However, if you do so, you must preface many of the file names in your CONFIG.SYS file with path names.

A Subdirectory for Every Software Package Many software packages use overlays, help files, device drivers, and other ancillary files. Such programs need to be able to locate these files. You can minimize confusion by giving each package its own directory. You will also reduce the risk of overwriting files with names like INSTALL.EXE and READ.ME, which may accompany several programs.

Keeping Your Organization Logical Your directory tree can greatly enhance your productivity if it helps you find your data files. Many authorities suggest making functional subdirectories children of the software packages that use them. For example, INVOICES, PAYROLL, and AP-AR might be children of a spreadsheet directory; LETTERS, MEMOS, and REPORTS might be children of a word-processor directory.

However, you can't always log onto a child directory from within a software package or call the software from the child directory. Even if your principal software packages are on your search path, you can reach your functional directories more easily if they are below the root directory rather than the software directories.

Keeping the data subordinate to the software may make sense if several people share your computer (perhaps a bookkeeper or accountant who does all the spreadsheet work and a secretary who creates the letters, memos, and reports). However, if you are the only person who uses your computer, it's much easier to find what you need if you group the files for a particular project in a single directory. Thus, one directory might hold invoices, memos, correspondence, project schedules, cost analyses, graphs, and so on. You can tailor your Path and Append statements to allow you to group your files by project and then create batch files for the rest. The directory in Figure 28 is a good example of what you should *not* do when organizing your directories. For further discussion of this point, see the Naming Subdirectories heading.

Making It Easy for DOS to Find What It Needs Finally, you want DOS to have easy access to the \DOS, \BATCH, and \UTILITY directories. (See Batch Files for more information about what goes in the \BATCH directory.) If these directories are on the search path, you can quickly execute the programs in them from anywhere in your system. This simplifies your housekeeping tasks.

Above all else, keep your system simple so that it will be easy to use and modify.

Nesting Subdirectories In principle, the only limitation on the degree of subdirectory nesting is the length of the path names you can type. For example, a DOS command line can hold only 132 characters, and some commands do not accept path names longer than 63 characters. You may have trouble reaching files in deeply nested subdirectories if path names referring to the files on the lowest subdirectory level exceed these limits.

You should usually nest subdirectories no more than two levels below the root; add the second level only if your directories are so full that you can't read them. Many popular utility programs, however, display a diagram of your tree and let you make any directory current by moving a

cursor to its name. If you use such a program, you may be able to navigate through directories nested three or more levels deep with little difficulty.

Naming Subdirectories Figure 28 shows another practice to avoid—using the same name for several subdirectories. If you really need so many subdirectories for your files, your software is probably not very efficient. Moreover, the file-cabinet analogy neglects one important point: The low-level files are created by your software. Your software can probably recognize its own files. Thus, they need not be in separate "folders." A decent database program can keep invoice files for four regions and four quarters in only sixteen files—or possibly in only one. If these sixteen files are in a single \INVOICES directory, as Figure 30 suggests, it's much easier to compare figures among regions and quarters.

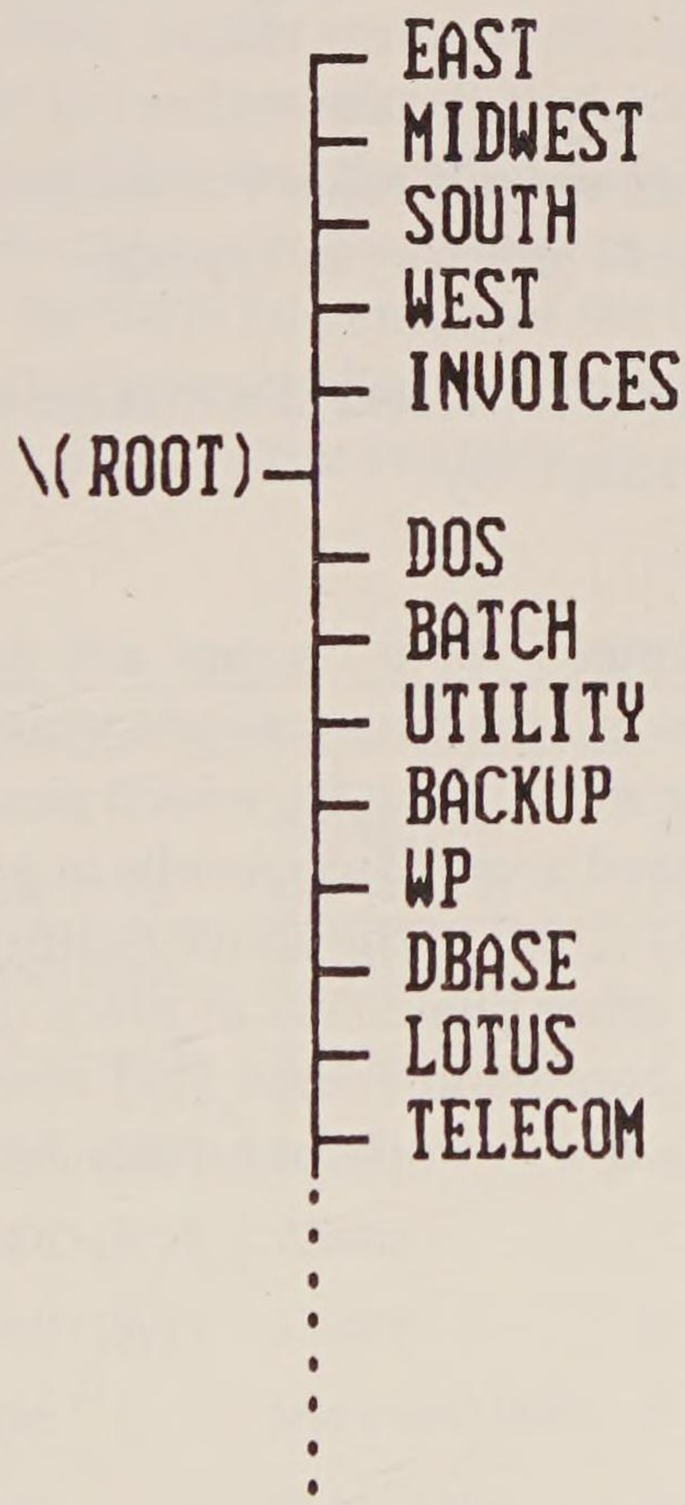


Figure 30: A sensible directory tree

Figure 30 depicts a better organization for the information found in the directories in Figure 28. The four regional directories are now children of the root, and they contain all the information for each sales region except the invoices. Again, this logical grouping makes comparisons easier.

Of course, this particular structure may not be the most useful structure for your business. In some cases, it might make more sense to group files by quarters so that comparisons can be made among regions for a given quarter. It all depends on the way you—and the organization you work for—do your work.

Subdirectory names follow the same rules as file names. Thus, each subdirectory can have a name up to eight characters long, plus a three-character extension. Giving extensions to your subdirectory is generally ill-advised since typing commands involving path names is difficult enough. However, if you give the same extension to all the subdirectory names (for example, .SD), you can easily get a list of subdirectories by issuing a command such as

```
DIR *.SD
```

If your subdirectory names do not have extensions, you can see the subdirectories to your current directory by typing

```
DIR *.
```

You may also see the names of some text files that have no extension, but it's a small price to pay for simplicity.

Tips

Displaying Directory Names Your prompt will display the name of the current directory if you use the prompt meta-string $p. Conventionally, this is followed by the meta-string $g, which generates the greater-than character. The command `prompt $p$g` results in your screen prompt reading `C:\REPORTS\MIDWEST\QTR1>` in the above example.

You can, however, use other characters in place of the greater-than symbol. For example, the command `prompt [$p]` results in the prompt `[C:\REPORTS\MIDWEST\QTR1]`. See Prompt for further details.

Disk Clusters

Overview

A *disk cluster* is the smallest amount of space on a disk that DOS can allocate when it writes a file, deallocate when it erases a file, or read when it reads a file. Thus, even a one-byte file occupies a full cluster of disk storage space.

DOS numbers the clusters on a disk consecutively, beginning with 2. Numbered clusters begin *after* the DOS reserved area, which includes the partition table on a hard disk, the boot record, two copies of the file allocation table, and the root directory (see Boot Record and Directories). Table 9 summarizes the number of disk sectors and bytes per cluster for common types of disks. The assignment of cluster numbers follows the DOS sector-numbering scheme rather than the ROM BIOS sector-numbering scheme.

Each cluster contains one or more *disk sectors* (see Disk Sectors). DOS groups sectors into clusters for reading and writing files to minimize the size of the file allocation table (see File Allocation Table). The file allocation table contains addresses for each cluster on the disk.

The DOS 4.0 version of the ChkDsk command displays the number of disk clusters (called "allocation units" in the ChkDsk report) on a disk, and the number available for storing data. (See ChkDsk.)

TABLE 9

Cluster Sizes for Common Types of Disks

DISK TYPE	NOMINAL CAPACITY	SECTORS PER CLUSTER	BYTES PER CLUSTER
SSDD (5¼")	160/180Kb	1	512
DSDD (5¼")	320/360Kb	2	1,024
DSHD (5¼")	1.2Mb	1	512
DSDD (3½")	720Kb	2	1,024
DSHD (3½")	1.44Mb	1	512
Hard	less than 16Mb	8	4,096
Hard	16 to 32Mb	4	2,048

D DISK CLUSTERS

If you use the DOS 4.0 version of FDisk to create disk partitions larger than 32 megabytes, the size of the clusters may be different from those described in Table 9.

Warnings If you have a 10-megabyte hard disk, and you upgrade from a version of DOS 2 to a version of DOS 3, you may find that your hard disk clusters now contain sixteen sectors (or 8,192 bytes). To find out if this is so, take the following steps:

1. Enter a Dir command, then note the number of bytes free on the last line.
2. Write a small file—1 or 2 bytes is sufficient.
3. Enter another Dir command, then note the number of bytes free on the last line.
4. Subtract the second number of bytes free from the first.

If the result is 8,192, your hard disk is allocated in sixteen-sector clusters. To correct this condition:

1. Do a complete backup of your hard disk because the following steps will erase all the data from it (see Backup and Hard Disks, Backing Up and Restoring).
2. Run FDisk (see FDisk).
3. Remove the current DOS partition.
4. Create a new DOS partition that takes up as much of the disk as your old one did. (You may change the size of the partition if you wish.)
5. Reboot from a diskette containing the version of DOS you want to use.
6. To transfer the operating system to the hard disk, enter the command
   ```
   format c:/s/v
   ```
7. Restore the backed up files to the hard disk; do not overwrite the new system files and COMMAND.COM (see Restore).

Follow the same procedure if you are upgrading a hard disk from DOS 2 to DOS 3 or DOS 3 to DOS 4 for the first time. This isn't absolutely necessary except with 10-megabyte hard disks, but it's a good idea with any size disk.

Tips For further details, see Disk Sectors; Diskettes, Structure of; Hard Disks, Structure of; File Allocation Table.

Disk Sectors

Overview A *disk sector* is the smallest amount of addressable storage space on a disk. Every type of diskette and hard disk addressed by DOS releases earlier than 4.0 uses a sector size of 512 bytes. (DOS 4.0 may use larger sectors on hard disk partitions that are larger than 32 megabytes.) If you use RAM disks, however, you can give them smaller or larger sector sizes.

DOS numbers the sectors on a disk consecutively, beginning with 0. The numbers are assigned first to the sectors on track 0 of side 0 (of a diskette) or head 0, cylinder 0 (of a hard disk partition). Numbering continues on track 0 of the second side of a diskette, and track 0 of subsequent heads on hard disks. Numbering then continues on track 2 of side 0 (cylinder 1 on a hard disk), and so on until all sectors are numbered. (Side 0, track 0 of a hard disk is reserved for the partition table.) These numbers are considered *logical* sector numbers.

The ROM BIOS has a different numbering scheme. The ROM BIOS uses an addressing scheme consisting of a side number, track number, and sector number. Sides or cylinders are numbered beginning with 0, tracks beginning with 0 and sectors beginning with 1. On each track on each side or cylinder, sector numbers begin with 1 again. These are considered *physical* sector numbers. Figure 31 illustrates the relationship between DOS and BIOS sector numbering. The numbers shown in the sectors are the DOS sector numbers.

Note that on hard disks, side 0, track 0 using the BIOS sector numbers is reserved for the partition table. The first partition thus actually begins on head 1 of cylinder 0. Except in DOS 4.0, any other partitions begin on head 0.

When reading or writing files, DOS works with a unit called the disk cluster, which may contain one or more sectors. (See Disk Clusters; Diskettes, Structure of; Hard Disks, Structure of; RAMDRIVE.SYS; and VDISK.SYS.)

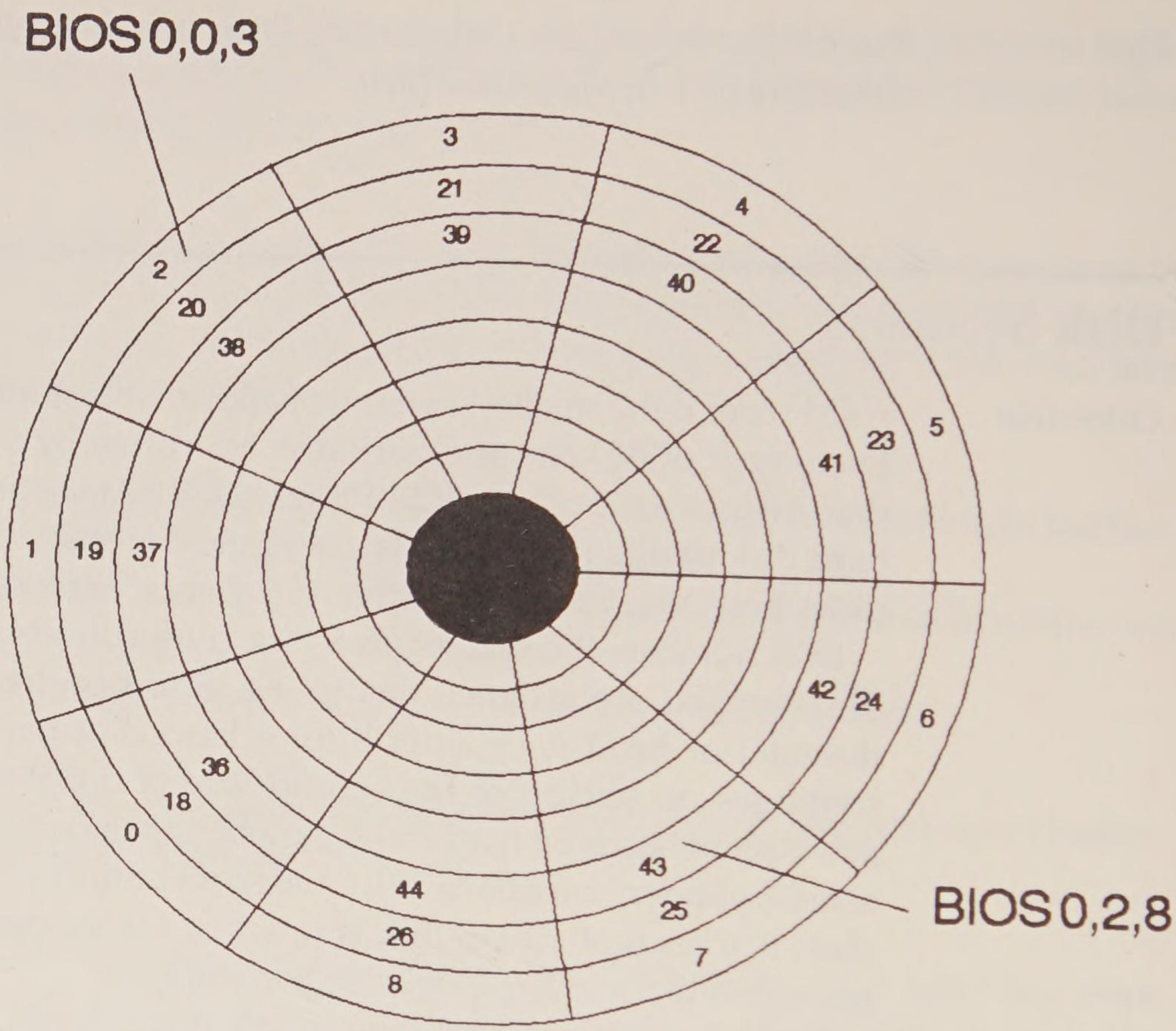


Figure 31: Disk sector numbering systems

DiskComp

Syntax

`diskcomp [drive] [drive] [/8][/1]`

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

DISKCOMP

D

Overview

The DiskComp command compares diskettes on a track-by-track basis. The command is most useful for comparing diskettes made with the DiskCopy command (see DiskCopy). You might think of DiskCopy as equivalent to the Verify option of the Copy command, but for diskettes instead of files.

Procedure

DiskComp is an external command. Using it requires having the DOS program DISKCOMP.COM on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:diskcomp a: b:
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\diskcomp a: a:
```

Diskettes to be compared must be of the same type. If you have one drive, DiskComp recognizes only drive A. If you have two drives, you may specify drives A and B. You can compare two 360K diskettes when one is in a 360K drive and the other is in a 1.2-megabyte drive. Similarly, you can compare two 720K diskettes when one is in a 720K drive and the other is in a 1.44-megabyte drive. If you are running the command from a hard disk, you *must* specify the drive, as in the above example, or the command will fail.

When you enter the command specifying two drives, the message

```
Insert FIRST diskette in drive A:

Insert SECOND diskette in drive B:

Press any key when ready . . .
```

appears. This gives you a chance to take the DOS diskette out of drive A if it's there and to replace it with the diskette you want to compare. DOS then reads as much of the diskette in drive A as memory will hold and compares it, track by track, with the diskette in drive B. If it finds a difference, a message of the form `Compare error, Side n, Track n`

appears, and the computer beeps. If the diskettes are the same, the message `Compare OK` appears after DOS has read both disks. Before the command terminates, it displays the message `Compare another diskette (Y/N)?`

To compare more diskettes, type `Y`; otherwise type `N`.

When you are using a single drive, DiskComp first displays the message

```
Insert FIRST diskette in drive A:

Press any key when ready . . .
```

When it has read as much of the diskette as memory will hold, it displays the messages

```
Insert SECOND diskette in drive A:

Press any key when ready . . .
```

These messages alternate until DOS has read both diskettes.

If you do not enter a drive name, DOS assumes the default drive. If the default drive is not a diskette drive, the program ends with an error message.

To interrupt the comparison, press Ctrl-Break. You may want to do this after several beeps. These beeps are an indication that the diskettes are not similar.

To compare single-sided diskettes or to compare only one side of double-sided diskettes, use the /1 switch. To compare eight-sector diskettes, or only eight sectors per track, use the /8 switch.

Warnings

DiskComp compares diskettes, not files. Therefore, if you copy all the files from one diskette to another with the Copy or Xcopy command, DiskComp will see the diskettes as different.

DiskComp recognizes only physical drives and logical drives created by DRIVER.SYS (see DRIVER.SYS). It does not recognize logical drives created by Assign, Join, or Subst.

In some DOS versions, DiskComp has exit codes that you can test with the ErrorLevel command (see ErrorLevel).

DiskCopy

Syntax

DiskCopy [*drive*] [*drive*] [*/1*]

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

DiskCopy copies a diskette to another diskette of the same type. It copies diskettes on a track-by-track rather than a file-by-file basis. If the target diskette is not formatted, DiskCopy formats it.

Procedure

DiskCopy is an external command; using it requires having the DOS program DISKCOPY.COM on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:diskcopy a: b:
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\diskcopy a: a:
```

Diskettes to be copied must be the same type. If you have one drive, DiskCopy recognizes only drive A. If you have two drives, you may specify drives A and B. You can copy a 360K diskette when the target diskette is in a 360K drive and the source is in a 1.2-megabyte drive. Similarly, you can copy a 720K diskette when the target diskette is in a 720K drive and the source is in a 1.44-megabyte drive. If you are running the command from a hard disk, you must specify the drive, as in the above example, or the command will fail.

When you enter the command specifying two drives, the message

```
Insert SOURCE diskette in drive A:

Insert TARGET diskette in drive B:

Press any key when ready . . .
```

appears. This gives you a chance to take the DOS diskette out of drive A if it's there and to replace it with the diskette you want to copy. DOS then reads as much of the diskette in drive A as memory will hold, and writes it, track by track, to the diskette in drive B. When the copy has been made DiskCopy displays the message

```
Copy complete

Copy another diskette (Y/N)?
```

To copy more diskettes, type Y; otherwise, type N.

When you are using a single drive, DiskCopy first displays the message

```
Insert SOURCE diskette in drive A:

Press any key when ready . . .
```

When it has read as much of the diskette as memory will hold, it displays the messages

```
Insert TARGET diskette in drive A:

Press any key when ready . . .
```

It then writes from memory to the target diskette. These messages alternate until the copy is complete.

If you do not enter a drive name, DOS assumes the default drive. If the default drive is not a diskette drive, the program ends with an error message.

To interrupt the copy, press Ctrl-Break.

To copy single-sided diskettes, or to copy only one side of double-sided diskettes, use the /1 switch.

Warnings

DiskCopy recognizes only physical drives and logical drives created by DRIVER.SYS (see DRIVER.SYS). It does not recognize logical drives created by Assign, Join, or Subst.

Tips

If you have a 1.2-megabyte drive and a 360K drive in your system, place the source diskette in the 1.2-megabyte drive when making a copy with DiskCopy. When you format a 360K diskette in a 1.2-megabyte drive, 360K drives may not be able to read it reliably. Similarly, if you have a 1.44-megabyte drive and a 720K drive, place the source diskette in the 1.44-megabyte drive.

If you have written to a diskette many times, files on it are likely to be fragmented. Consequently, DOS takes longer to read them. DiskCopy does not affect fragmentation. To make a copy with the files unfragmented, use the Copy or Xcopy command (see Copy and Xcopy).

In some DOS versions, DiskCopy has exit codes that you can test with the ErrorLevel command (see ErrorLevel).

To check the accuracy of a copy, use the DiskComp command (see DiskComp).

Diskettes, Structure of

Overview

DOS can read and write to many types of diskettes. However, all diskettes formatted by DOS are set up in fundamentally the same way. DOS reserves certain areas of a diskette for its own purposes, and these areas always appear in the same location, in the same order on the disk.

Diskettes are laid out as a series of concentric rings called *tracks*. Single-sided diskettes have tracks numbered consecutively beginning at the outer edge with 0. Double-sided diskettes have two sides, numbered 0 and 1. Tracks are still numbered consecutively from the outer edge, but each track is considered as beginning on side 0 and continuing on side 1.

Each track is divided into *sectors* holding 512 bytes of data (see Disk Sectors). Some of the sectors on track 0 are reserved by DOS for its own purposes.

The very first sector is the boot record (see Boot Record). This is a short program that starts the computer. Two copies of the file allocation table follow it immediately (see File Allocation Table). The root directory appears next (see Directories).

The size of the file allocation tables and root directory varies with the capacity of the diskette. Each root directory sector can hold sixteen entries. Table 10 summarizes the size of the DOS reserved areas for various

types of diskettes. Figure 32 shows the layout of these areas on a double-sided, double-density 5¼" diskette. Sector numbers are shown in the DOS sector-numbering scheme.

The rest of the diskette is used for storing your data. It is divided into clusters of one or two sectors, which are numbered consecutively beginning with 2 (see Disk Clusters). Single-sided and high-density diskettes have one sector per cluster. Double-sided, double-density diskettes have two sectors per cluster.

If the disk is a system disk (that is, one that you can start your computer with), the hidden system files IBMBIO.COM and IBMDOS.COM, or IO.SYS and MSDOS.SYS, occupy the sectors immediately following the root directory. If you try to make a disk bootable with the Sys command and see the message

```
No room for system on destination disk
```

this indicates that some of the data sectors following the root directory are already occupied by other files (see Sys) or that the version of DOS whose system files you want to transfer is incompatible with the one that formatted the disk.

Tips For related information see Boot Record, Directories, Disk Clusters, Disk Sectors, File Allocation Table, and Hard Disks, Structure of.

TABLE 10 **DOS Reserved Area of Various Diskette Types**

DISK TYPE	NOMINAL CAPACITY	FAT SECTORS	DIRECTORY SECTORS	DIRECTORY ENTRIES
SSDD (5¼")	160/180K	2	4	64
DSDD (5¼")	320/360K	4	7	112
DSHD (5¼")	1.2Mb	14	14	224
DSDD (3½")	720K	10	7	112
DSHD (3½")	1.44Mb	18	14	224

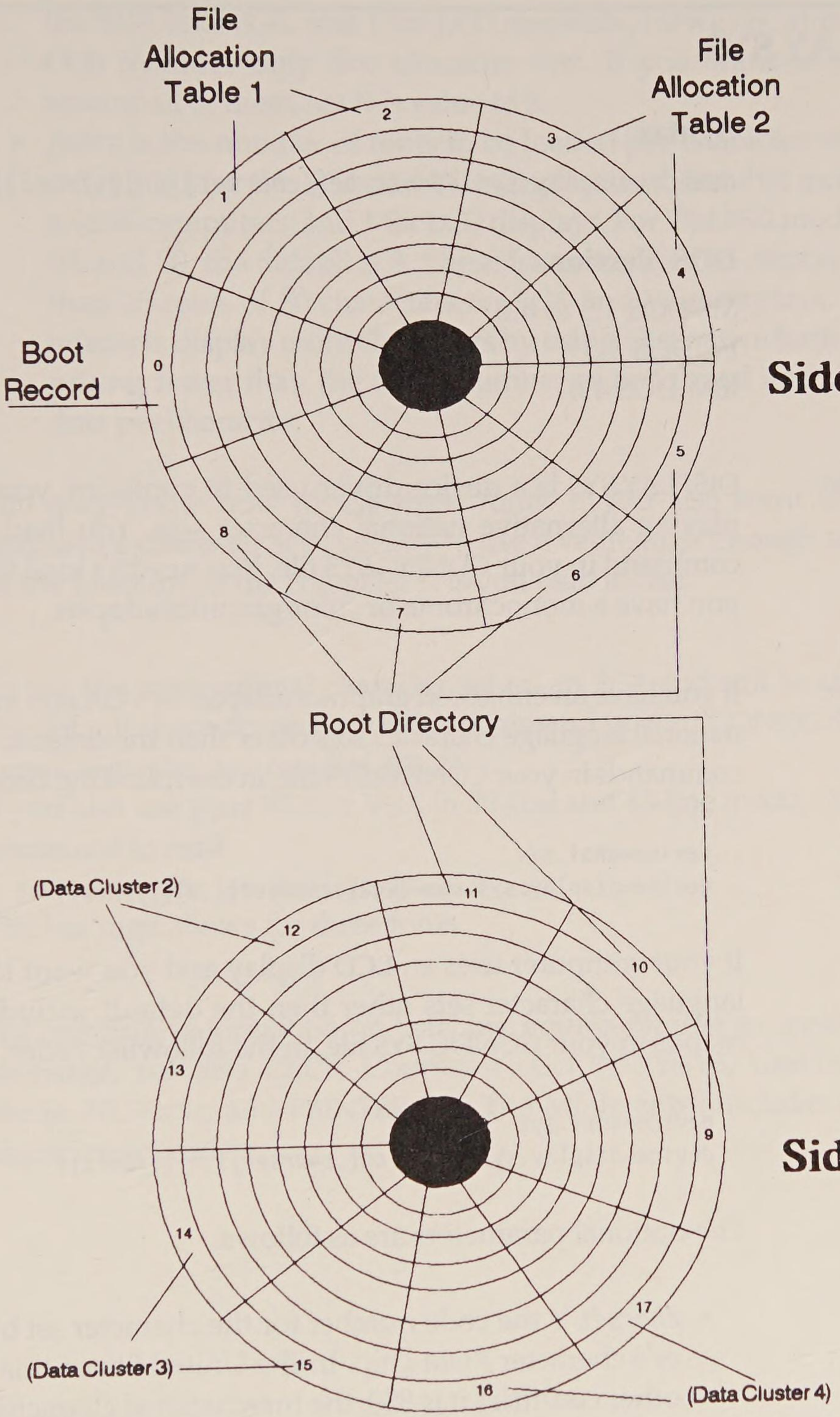


Figure 32: The DOS reserved areas on a DSDD diskette

DISPLAY.SYS

Syntax

device=display.sys con=(*type*[,*charset*][,*n*][,*fonts*])

DOS Versions

MS-DOS 3.3 to 4.0
PC-DOS 3.3
IBM DOS 4.0

Overview

DISPLAY.SYS is a device driver used to configure your monitor for displaying alternative national character sets. You load it via a Device= command in your CONFIG.SYS file. You needn't load this device driver if you have a monochrome or color/graphics adapter.

Procedure

If you have an enhanced graphics adapter or VGA and you want to display national language character sets other than the default, include these two commands in your CONFIG.SYS file, in the following order:

```
device=ansi.sys
device=display.sys con=(ega[,charset][,n][,fonts])
```

If your computer uses an LCD display and you want to display national language character sets other than the default, include these two commands in your CONFIG.SYS file, in the following order:

```
device=ansi.sys
device=display.sys con=(lcd[,charset][,n][,fonts])
```

The optional parameters are as follows:

- *charset* is the code number for the character set built into adapter's character ROM chip. In the United States this is 437. In most other countries it is 850, the international character set.
- *n* is the number of additional character sets you want to use. Acceptable values range from 0 to 12, with the default being 2 for

the EGA and VGA, and 1 for LCD monitors. However, at present DOS includes only five character sets. If you want to switch among all of them, set this value to 5.

- *fonts* is the number of fonts to be loaded per character set. The default is 2 for the EGA and VGA displays and IBM PS/2 model 25 and 30 computers, and 1 for LCD displays. For IBM PS/2 models 50, 60, and 80, the default is 3. These fonts are used to display more than 25 rows of 80 characters on EGA and VGA displays. These adapters display more characters by using alternative fonts, comprising fewer than the usual number of horizontal and vertical dots per character.

You may know how to use these fonts. If you also want to use the alternative character sets, be sure to use a value high enough to include all the fonts for all the character sets you plan to use.

Examples

To use the international character set on an EGA and still be able to use the default character set, enter this command in your CONFIG.SYS file:

```
device=display.sys con=(ega,850,2)
```

If you also use your EGA or VGA in 35-line and 43-line mode, change the command to read

```
device=display.sys con=(ega,850,2,3)
```

The last digit allows for three fonts.

Tips

For complete information on teaching your computer to speak another language, see also ChCP, Country=, COUNTRY.SYS, GrafTabl, Keyb, Mode, NLSFunc, and PRINTER.SYS. The Mode entry includes a table of character-set codes.

DOSShell

Syntax

dosshell

DOS Versions

MS-DOS 4.0
IBM DOS 4.0

Overview

IBM DOS 4.0 and most releases of MS-DOS 4.0 include a utility known as a *DOS shell*. Generally, a DOS shell gives you a menu interface to DOS in place of the command line. The shell included in DOS 4.0 is a collection of utilities invoked through a batch file called DOSSHELL.BAT. You can customize the shell by adding and deleting commands and parameters in this file.

DOSShell gives you a graphics-based interface that works in either text or graphics mode. In its full configuration, it includes a menu system (called Start Programs) from which you can start any or all of your programs, a file system for doing all your housekeeping, and some utilities for maintaining the shell itself. You can add items to the start programs menu or add a second level of menus.

By deleting some of the options on the main menu, and installing passwords in the menu options that invoke programs, you can add a limited measure of security to your system. However, this protection is so limited that it can be broken simply by booting the system from a diskette that doesn't include a call to the DOSSHELL batch file in its AUTOEXEC.BAT file.

The DOSShell program uses the following files from the DOS 4.0 system disk:

FILE NAME	COMMENTS
DOSSHELL.BAT	The batch program that sets up the shell
DOSUTIL.MEU	The DOS Utilities menu file
SHELL.CLR	The default color file
SHELL.HLP	The help file
SHELL.MEU	The shell's opening menu
SHELLB.COM	The memory-resident loader program
SHELLC.EXE	The executable program

If you use the File/Associate command (described under Warnings), you create an additional file called SHELL.ASC. This file contains a list of the extentions used by various programs for their data files.

Procedure/ Examples

Configuring the Shell When you install DOS 4.0 using the Select command, by default it builds a file called DOSSHELL.BAT in your DOS directory. DOS 4.0 also automatically includes a call to this file at the end of the AUTOEXEC.BAT file it creates for you. You can make changes even before you complete the installation, if you select `2. Review, change, or add installation choices` when given that option while using the Select program. Eventually, you will be presented with a screen like the one in Figure 33. You can customize the shell by adding or deleting parameters from the line shown between the symbols [and >. You can scroll the text along this line, and enter a total of 115 characters. Select the parameters from Table 11.

If you have already installed DOS 4.0, you can still customize the shell by adding or deleting these parameters from the Shellc command in DOSSHELL.BAT. A sample version of DOSSHELL.BAT follows. Select actually writes the file all in uppercase, but since the convention established for this book requires programs to be in lowercase, I'm presenting it that way. It's not at all clear how most of this file works, and the commands

```
                    DOS SHELL Parameters

SELECT has set these values for you (if any).  Press
Enter to accept these values or type in new values.

SHELL parameters . .          [/MOS:PCIBMDRV.MOS/TRAN/COLOR/DOS>

Enter  Esc=Cancel  F1=Help
```

Figure 33: Select's DOSShell parameters screen

TABLE 11 **DOSShell Shellc Parameters**

PARAMETERS	COMMENTS
Functional Modules	
/date	Displays the current date and time on all DOSShell screens.
/dos	Installs the File System. Without this parameter, you can use only the Start Programs screen.
/exit	Permits exit from the shell to DOS.
/maint	Lets you add, change, delete, move, copy, and reorder options on the Start Programs screen.
/menu	Installs the Start Programs screen. Without this parameter, you can use only the File System.
/prompt	Allows access to a DOS command line. Without this parameter, the Command Prompt option on the Start Programs screen will do nothing (you can delete it).
/color	Allows you to change the colors in which DOSShell screens appear. Without this parameter, the Change Colors option on the Start Programs screen will do nothing (you can delete it). Again, even if you delete this parameter, the option will still appear on Start Programs screen unless you delete it.
/snd	Includes sounds (beeps).
Mouse Options	
/com2	Looks for a mouse on COM2 instead of COM1.
/lf	Sets up the mouse as a left-handed mouse.
/mos:pcibmdrv.mos	Installs the mouse driver for the PS/2 mouse (default).
/mos:pcmsdrv.mos	Installs the mouse driver for the Microsoft serial mouse, model 039–099.
/mos:pcmspdrv.mos	Installs the mouse driver for the Microsoft parallel mouse, model 037–099.
Display Options	
/co1	Installs the shell in EGA mode 10 (640 × 350 16-color graphics).
/co2	Installs the shell in VGA mode 11 (640 × 480 two-color graphics).
/co3	Installs the shell in VGA mode 12 (640 × 350 16-color graphics).
/text	Installs the shell in text mode.

Table 11 continues

PARAMETERS	COMMENTS
Modes of Operation	
/mul	Establishes multiple buffers for the File System and for directories.
/b:*n*kb	Specifies the size of the buffers in kilobytes.
/#swap	An analogue of the FastOpen command, saving directory and file information in a swap file on a hard disk, rather than in memory, for use when reloading the shell. If you have to reboot before exiting gracefully from DOSShell, the swap file will remain on disk.
/tran	Installs the shell as a transient program, rather than a TSR. When this option is installed, all but about 3K of the shell unloads itself from memory when you run a program or command within the shell, invoke the Command Prompt option, or exit from DOSShell entirely. If you omit this option, the entire program remains in memory, occupying about 100K of RAM.
File Options	
/meu:*filename*.meu	Specifies the name of the file containing the options for the Start Programs screen. If you use the /menu parameter, one such file name parameter must be included. The default is SHELL.MEU.
/clr:*filename*.CLR	Specifies the name of the color to file use. The default is SHELL.CLR.

won't work if you type them at a DOS prompt. However, the only part you need concern yourself with is the parameters to the Shellb and Shellc commands:

```
@c:
@cd\dos
@shellb dosshell
@if errorlevel 255 goto end
:common
@shellc /mos:pcibmdrv.mos/tran/color/dos/menu/mul/snd
  /meu:shell.meu/clr:shell.clr/prompt/maint/exit/swap/date
:end
@break=on
```

By default, Select configures the shell to use the IBM PS/2 mouse, the highest screen resolution available in your system, and a transient mode of operation. If you have color, it includes the option that allows you to change screen colors. It also includes all the functional modules that the shell can use:

- Access to a command-line prompt, from which you can return to the shell by typing `exit`
- The file system
- The ability to add, change, and delete items from the shell's menus
- Multiple buffers for the file system directory and file buffers (equivalent to the FastOpen command)
- The Start Programs screen, which is the base on which you build your menu system
- The ability to leave the shell entirely and use a DOS prompt
- Sound (beeping on errors)
- A display of the date and time on all shell screens

If your system cannot display graphics, the shell will run in text mode. You can force the shell to run in text mode by adding the /text parameter. It's a bit less impressive in text mode, but it runs faster, because it takes less time to update a text screen than a graphics screen.

If you already have a mouse driver in your CONFIG.SYS file, using one of the mouse drivers supplied with DOSShell will disable the mouse in other applications. However, most mouse drivers work perfectly well with the shell. Therefore, you should leave out or delete the shell's mouse driver file if you want to use the mouse with any other applications that require a mouse driver.

Using the Shell Figure 34 shows the first display screen as it appears by default. This is the Start Programs screen for the Main Group. Note that the menus are on a bar at the top of the screen. At the center of the screen is a list of items constituting the Main Group. At the bottom of the screen is a help line. You can get context-sensitive help at any time by pressing F1 or clicking the mouse on the F1 on the menu bar.

You can use the shell either with or without a mouse. To use the mouse, you point to the item you want to select, and click. To use a menu, point to the menu name, click, and the menu will open. Some menus and dialog boxes show key names on the screen. You can simulate the action of these keys by clicking on their names on the screen. You select files in

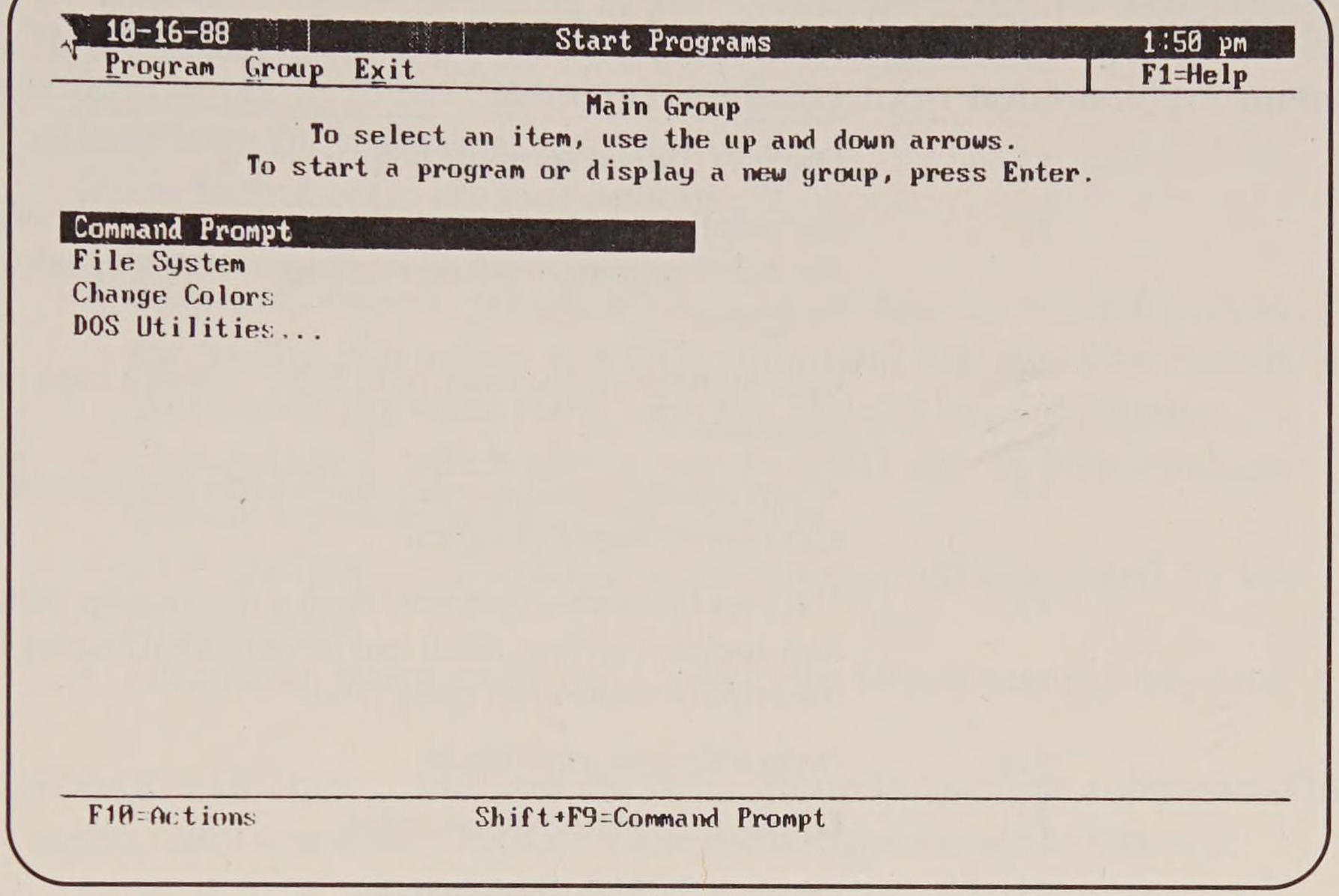


Figure 34: The DOSShell Main Group

the File System in the same manner. When selecting an item from a list, either in the Main Group or in a submenu, however, you must double-click to activate the item.

If you don't have a mouse, you can access the menus from the keyboard. Press F10, then choose a command either by moving the highlight to it with the cursor keys and pressing Return, or by pressing the key whose name is highlighted or underlined. Items that are not available will have their names displayed in a paler shade in graphics mode, or have their command key replaced by an asterisk in text mode. You move between parts of the screen—from the menus to the list, for example—by pressing the Tab key, or the Shift-Tab combination to move in reverse. Table 12 summarizes the keys that have consistent functions throughout DOSShell.

Using the Start Programs Screen Figure 35 is a schematic diagram of the Start Programs menu. As you can see, you can start any item displayed in the central part of the screen, or add, change, or delete programs or groups in the Main Group. When you add a group, you create a submenu. In addition, you can copy a program from one group to another, or reorder the names of the items in the group. To move a

TABLE 12 **DOSShell Command Keys**

KEYS	COMMENTS
F1	Displays context-sensitive help.
F2	Records the information you type in an entry field when customizing the Start Programs module; completes a copy operation in Start Programs.
F3	Exits to the previous level, or to DOS; cancels a copy operation in Start Programs.
F4	When creating menus, enters the marker that separates commands you enter to start a program.
F9	Displays the key assignments from within a help panel; when viewing files, toggles between ASCII and hexadecimal display formats; restores the original value in an entry field.
F10	Accesses menu commands.
F11	Displays the Help System index.
Alt-F1	Displays the Help System index.
Shift-F9	Loads a secondary command processor to give access to a DOS prompt.
Return	Confirms or completes a selection.
Esc	Cancels a selection or action.
PgDn	Scrolls to the next page of information.
PgUp	Scrolls to the previous page of information.
Space bar	Selects or deselects a file name.
Tab	Moves the selection cursor among the the different areas of the File System screen.

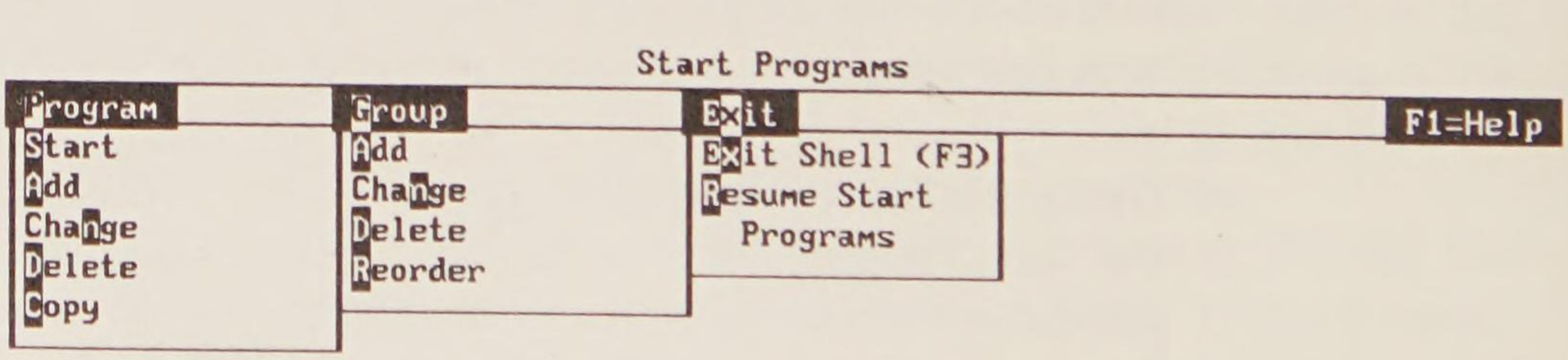


Figure 35: The DOSShell Start Programs menu

program from one group to another, first copy the program to the target group, and then delete it from its original group. We'll look more closely at these commands in the section on creating a menu system later in this entry.

In the Main Group are four options:

- *Command Prompt*, which loads a secondary command processor, giving you access to a DOS command line (see Command). You can do the same thing with the Shift-F9 key combination.
- *Change Colors*, which allows you to select one of four combinations of screen colors on a color system.
- *DOS Utilities . . .*, which is a secondary menu, denoted by the ellipsis mark.
- *File System*, which gives you access to the file-management system.

The DOS Utilities . . . entry in the main group is a sample submenu. The menu names, and the DOS commands they invoke, are as follows:

MENU	COMMAND
Set Date and Time	Date, Time
Disk Copy	DiskCopy
Disk Compare	DiskComp
Backup Fixed Disk	Backup
Restore Fixed Disk	Restore
Format	Format

(See the entries for each command for more information.) Most of these commands use the command language to include default values, which you can change. You can use these menu commands as models for your own menus.

Using the File System The file system allows you to:

- Run programs
- Print files (if you have PRINT.COM installed in memory; see Print)
- Copy, move, rename, or delete files individually, by wildcard relationships, or in groups you select, even if the files are in different directories
- View files in an ASCII or hexadecimal format

- Change file attributes (see File Attributes)
- Create, delete, or rename directories
- Associate file name extensions with the programs that create those types of files, so that you can invoke a program by double-clicking on one of its data files
- View file names in two directories at once; the directories may even be on different disks

Figure 36 presents a schematic diagram of the File System menu. The keys that activate the various menu commands are shown in inverse video. Those commands that have secondary commands appended to them, or that are followed by ellipses, display dialog boxes in which you make your selections. Default selections are marked by an asterisk.

Selecting File System on the Start Programs screen brings you to the File System screen, which appears in Figure 37. As you can see, the screen comprises several distinct areas. The default arrangement, which is called the Single File List, is shown. There are five areas, four of which are selectable: the menu bar, the disk drives, the directory tree, and the file list. The fifth area, showing the Current Directory window, changes when you select a directory from the directory tree. As in the Main Group, you move among the selectable areas with the Tab key or the mouse.

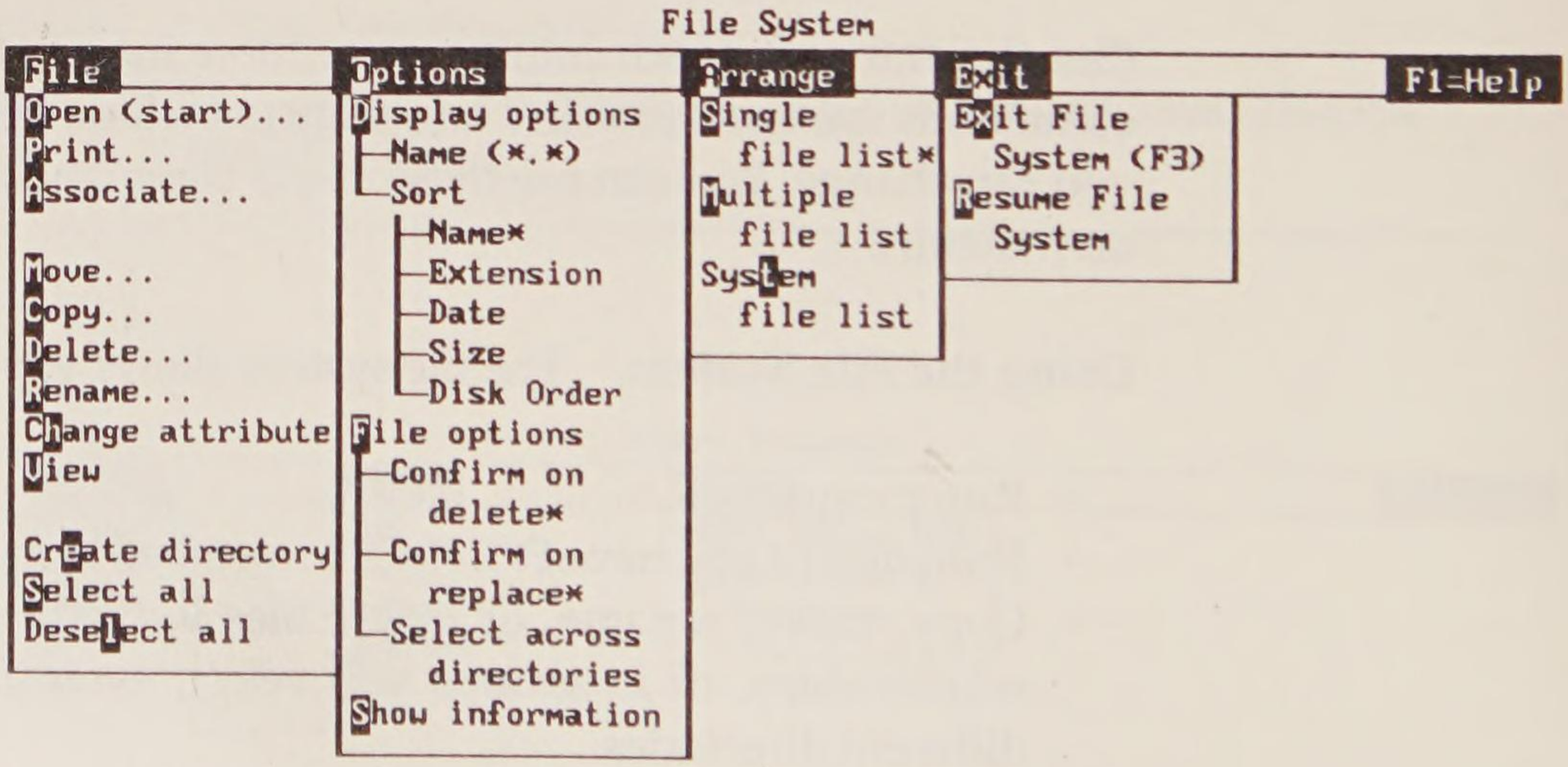


Figure 36: The DOSShell File System menu

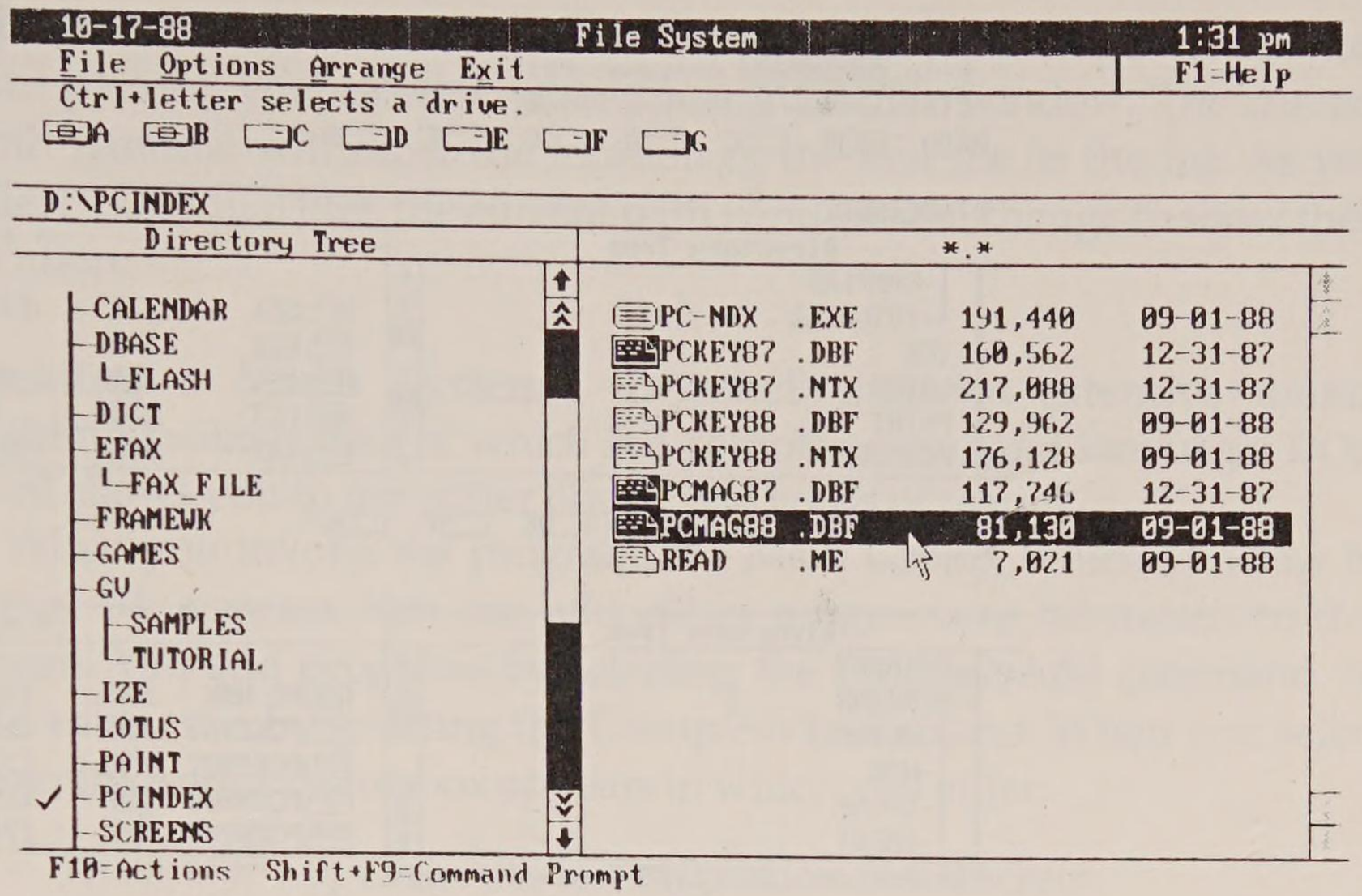


Figure 37: The File System Single File List

In graphics mode, a check mark indicates the selected directory, and the colors are inverted in the icons next to selected files. The icon for executable files is a box with three lines on it. All other files are marked by page of text with the corner folded. In text mode, a small arrowhead marks the selected directory and any selected files. You select disk drives, directories, or files with the cursor keys or the mouse when the selection bar is in the appropriate window.

You can choose one of two other arrangements from the Arrange menu. Multiple File List, shown in Figure 38, gives you a second drive, tree, and file window to let you work with two directories at once. System File List eliminates the tree window, and instead displays a window of file and disk statistics. You can display this window, which appears in Figure 39, no matter which Arrange option you select, using the Options/Show Information command. (The fuzzy gray area in the figure is the mouse cursor in text mode.)

The System File List is especially useful for locating files by name. You select a file name with the Options/Display Options/Name command. Then you turn on the Options/File Options/Select Across Directories

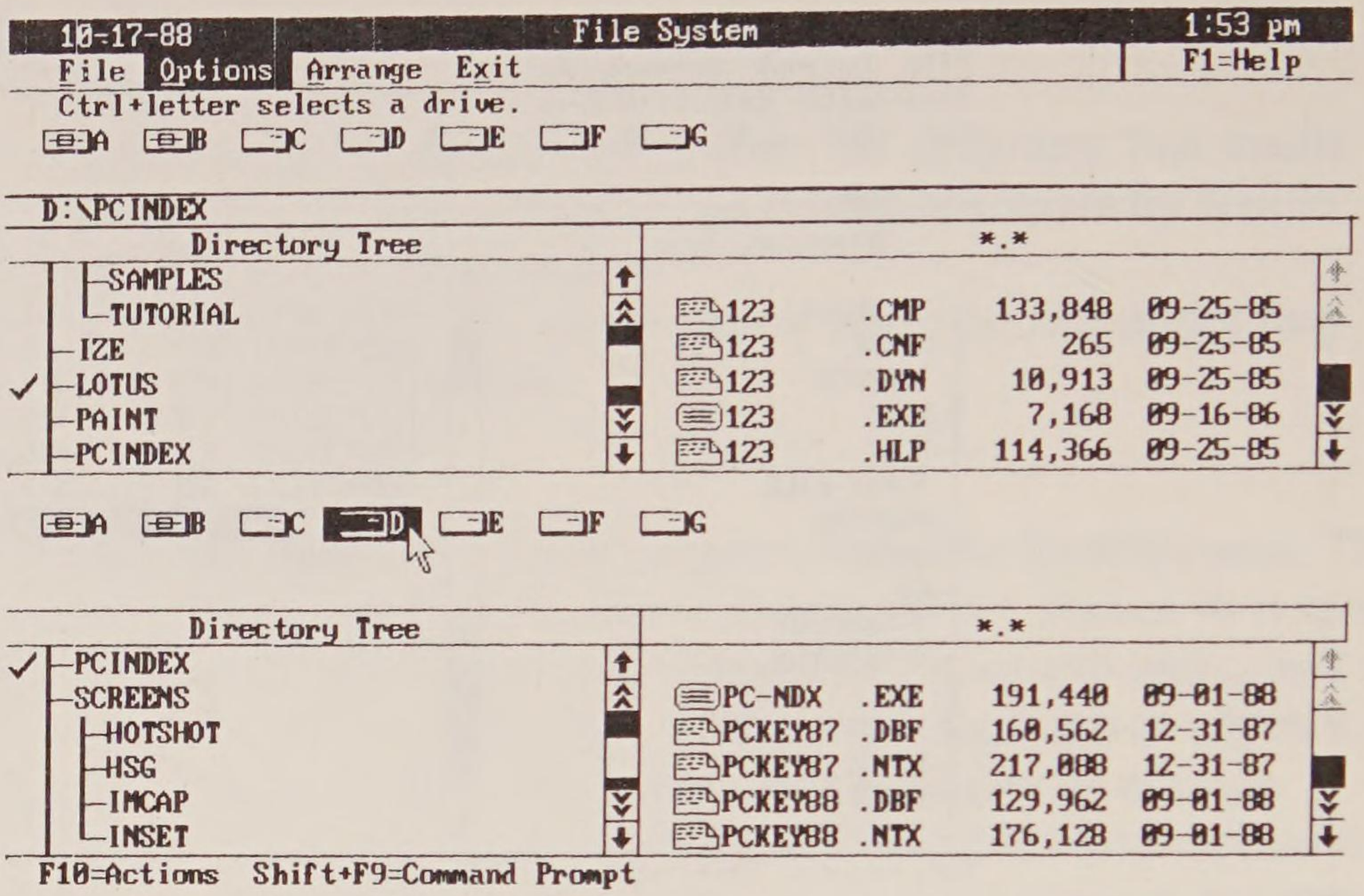


Figure 38: The File System Multiple File List

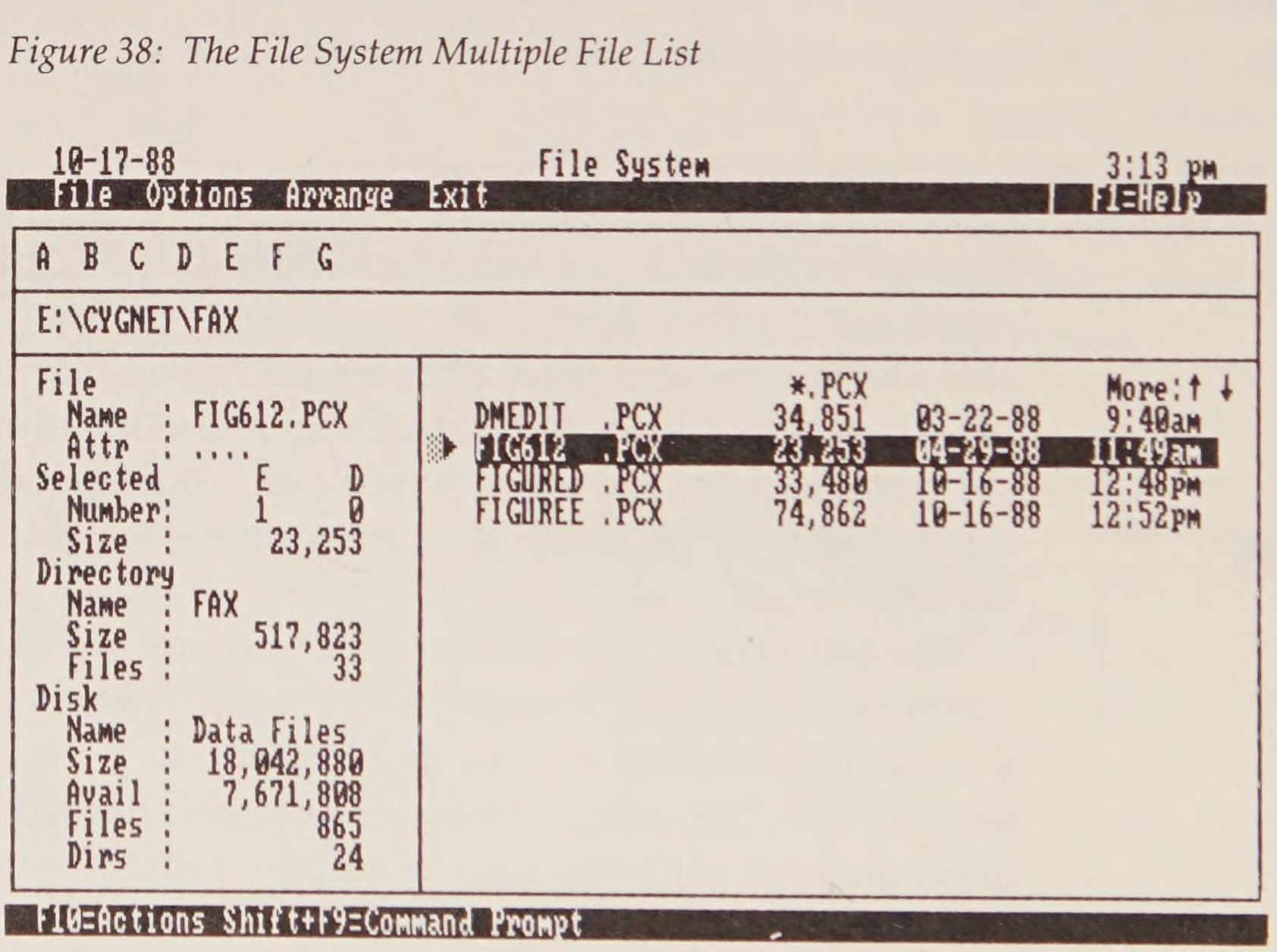


Figure 39: The File System System File List

option. All files anywhere on the disk that match the file name, or wild-card pattern, you entered, will appear in the file window. The current path window will show the location of the first file in the list. As you select individual files, the current path window will change to show their location.

Creating a Menu System DOSShell includes extensive menu-building facilities, one of which is a complex command language. DOS-Shell allows you to use either one or two levels of menus.

When you invoke the program, the Main Group, which you saw in Figure 34, appears. You can add either programs or submenus to this screen. You add programs by selecting the Program/Add command, or add submenus by selecting the Group/Add command. When you select Program/Add, a dialog box appears in which you enter:

- The text to appear in the central portion of the screen
- The commands with which to invoke the program (including special DOSShell commands)
- A password of up to eight characters
- The text for a help dialog box

The last two items are optional, but you cannot install programs without the first two. If you need more than one command to invoke a program, separate the commands by pressing F4, not Return. Press F2, or click the F2 button on the screen, to save your work.

To add a submenu, select Group/Add, and enter a file name for a file to hold the items in the group. The extension .MEU is supplied automatically. Figure 40 shows the Main Group with several submenus added. As you can see from the menu that is open, once you have installed programs or submenus, you can later:

- Delete them with the Delete command
- Edit the commands that invoke them with the Change command
- Change the order in which they appear on the screen with the Group/Reorder command
- Include programs in more than one group with the Copy command
- Move programs from one group to another with the Program/Copy and Program/Delete commands

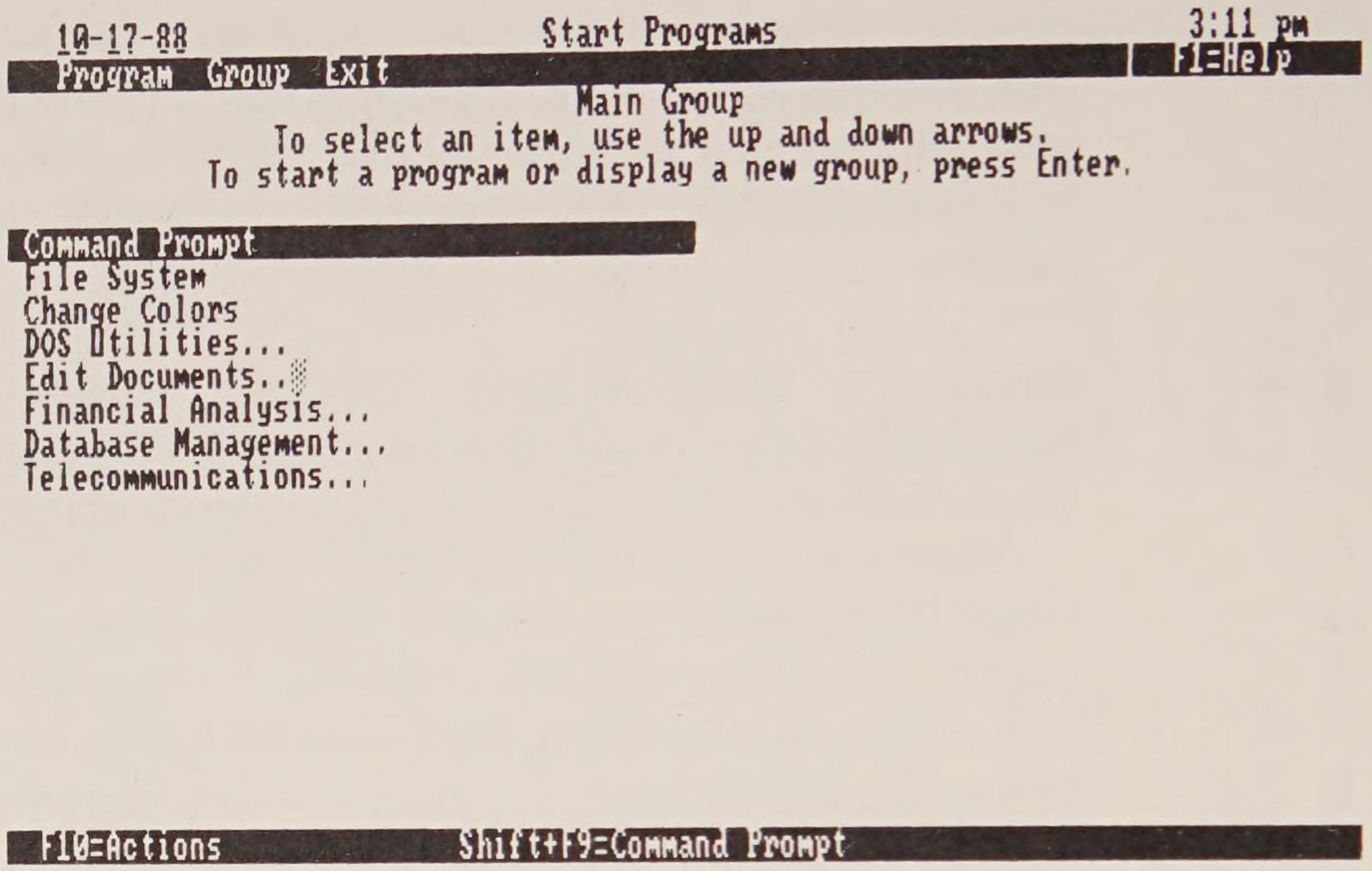


Figure 40: DOSShell with several submenus added

To add programs to a submenu, select the name of the group. You will then see the message `Group is empty`.

You can then add programs to the submenu the same way you would to the Main Group. When you are working in a group other than the Main Group, the only Group command available is Reorder.

With DOSShell's special command language and two levels of passwords you can exercise a fair amount of control over the way users invoke an application, what applications are available to them, and what files the application can open. The commands are summarized in Table 13.

The dialog box for adding a program to a DOSShell menu appears in Figure 41. (The Add Group, Change Program, and Change Group dialog boxes are virtually identical to it.) Notice that there is only a single line on which you can enter program commands. You can scroll along this line either by holding down the mouse button while the mouse cursor is over one of the markers at the ends of the field, or with the cursor keys. The entry fields, like all DOSShell entry fields, is in typeover mode, but you can switch to insert mode by pressing the Ins key. Because you can work on only a single line, and can see only 28 characters at a time, it's quite difficult to get an overview of what you are doing, or to keep track of what you've done.

TABLE 13 **DOSShell Special Commands**

COMMAND	EFFECT
[ ]	Displays the default prompt in the prompt panel.
[%*n*]	Accepts a parameter entered in the parameter field. *n* may have any value from 1 to 10.
[/t"*title*"]	Displays the title *title*. A title can be up to 38 characters long.
[/i"*instruction*"]	Displays the instruction line *instruction*. The instruction line can be up to 38 characters long.
[/p"*prompt*"]	Displays the prompt *prompt*. It can be up to 18 characters long.
[/d"*default*"]	Displays a default value on the parameter- entry line (need not be a file name); user can edit or delete the default value.
[/d"%*n*"]	Displays as a default the previously entered value assigned to %*n*.
[/r/d"*default*"]	Displays a default value, but clears it when an alphanumeric key is pressed; similar to [/d"*default*"], but the value displayed automatically goes away if the user begins to enter new text.
[/l"*n*"]	Defines a maximum length for the entered value; without an /l command, the entry field will be 127 characters long.
[/m]	Checks whether the file name entered already exists; if not, rejects the entry.
[/c"%*n*"]	Reuses the value assigned to %*n* in a previous command.
[/f]	Checks for the existence of the specified file (which may include a path name) before executing the program. If the file does not exist, returns to the shell.
/#	Automatically inserts the name of the drive from which you started the shell.
/@	Automatically inserts the full path from the root directory of the disk from which you started the shell to the current directory.

There are tricks to entering the special commands. Most important is the relationship between the brackets, the quotation marks, and the separators. As noted, you separate commands with the F4 key, which enters a graphics character between commands. You can enter a series of DOS

commands, separated by F4 characters, as if they were a batch file. However, DOSShell handles replaceable parameters differently from the batch language (see Batch Files). This is because it gives the user a dialog box (shown in Figure 42) in which to enter the parameters.

Suppose, for example, you want to install a utility called LOOK.COM. It will open a file if the file name is given as a parameter. If you were writing a batch file to call this program, it might contain the single line

```
@c:\tools\look %1 %2 %3
```

```
                    Add Program
Required

 Title . . . .   [                                    >

 Commands  . .   [                                    >

Optional

 Help text . .   [                                    >

 Password  . .   [          ]

Esc=Cancel    F1=Help    F2=Save
```

Figure 41: The Add Program dialog box

```
                Program Parameters

Type the parameters, then press Enter.

   Parameters . .   [                                 >

<-┘=Enter    Esc=Cancel    F1=Help
```

Figure 42: The Program Parameters dialog box

To simulate this command in DOSShell, you need to enter

```
[%1 %2 %3]||c:\tools\look %1 %2 %3
```

(The double bar is the character inserted by the F4 key.) When you select this program from the group on the screen, you see the default Program Parameters dialog box, shown in Figure 42.

However, you can make the dialog box much easier to use. Notice that the Program Parameters box has three separate text areas: the title (Program Parameters), the instructions (Type the parameters, then press Enter), and the prompt (Parameters...). You can change each of these with the /t, /i, and /p commands, respectively. However, you must include the changes within a single set of brackets for each dialog box. By careful use of the special commands, you can create a separate dialog box for each parameter, each with helpful information.

Let's assume the program you're invoking is a bit more complex than the previous example. In this instance, you want to change the title of the box, and explain what is to be entered. You might then, for example, enter the commands

```
[%1/t"The Editor"/i"Enter the name of the file to open"/p"File name: "/l"12"]||
[%2/t"The Editor"/i"If not in current directory enter path"/p Full path: "]||
[%3/t"The Editor"/i"Enter the Profile File to use"/l"12"/p"Profile file: "]||
cd %2||
c:\edit\edit %1 /p:%3
```

For the sake of readability, these commands appear here on separate lines. In DOSShell, they are all on a single line, only part of which is visible, as Figure 43 shows. You can see from this example that all the special commands relating to a single dialog box must be enclosed in a single set of brackets. The replaceable parameter symbol must be inside the same set of brackets, but not enclosed in quotation marks. You might want to add some help text telling the user not to add a backslash to the end of the path name, because it already is included in the command. When an error message appears, the user can press F1 for further assistance. The first of the dialog boxes is shown in Figure 44.

If you have already written batch files that you want to use, you can install them in the shell instead of writing complex sequences of commands. However, you won't have access to as much on-screen help. The

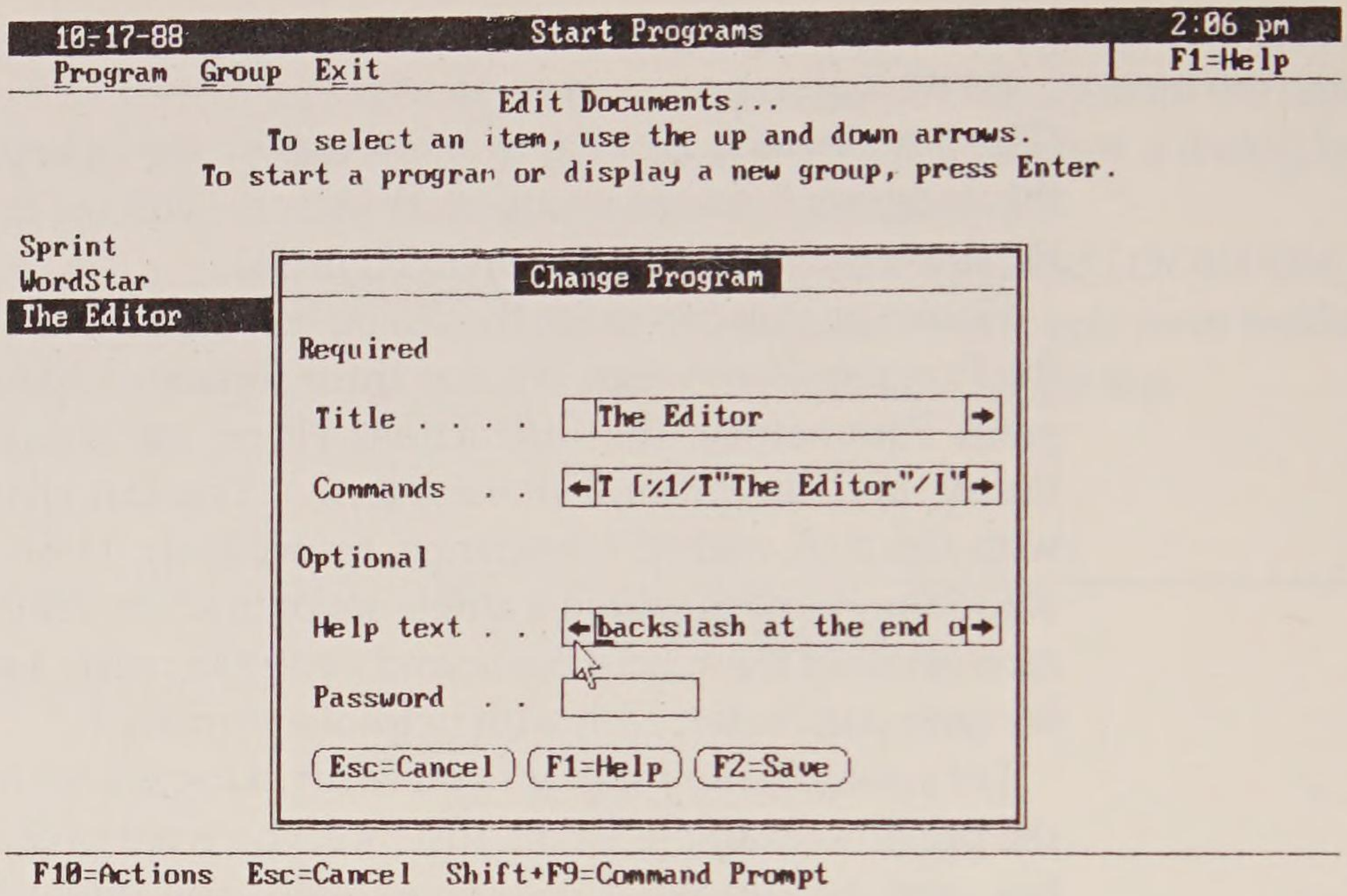


Figure 43: Setting up a program

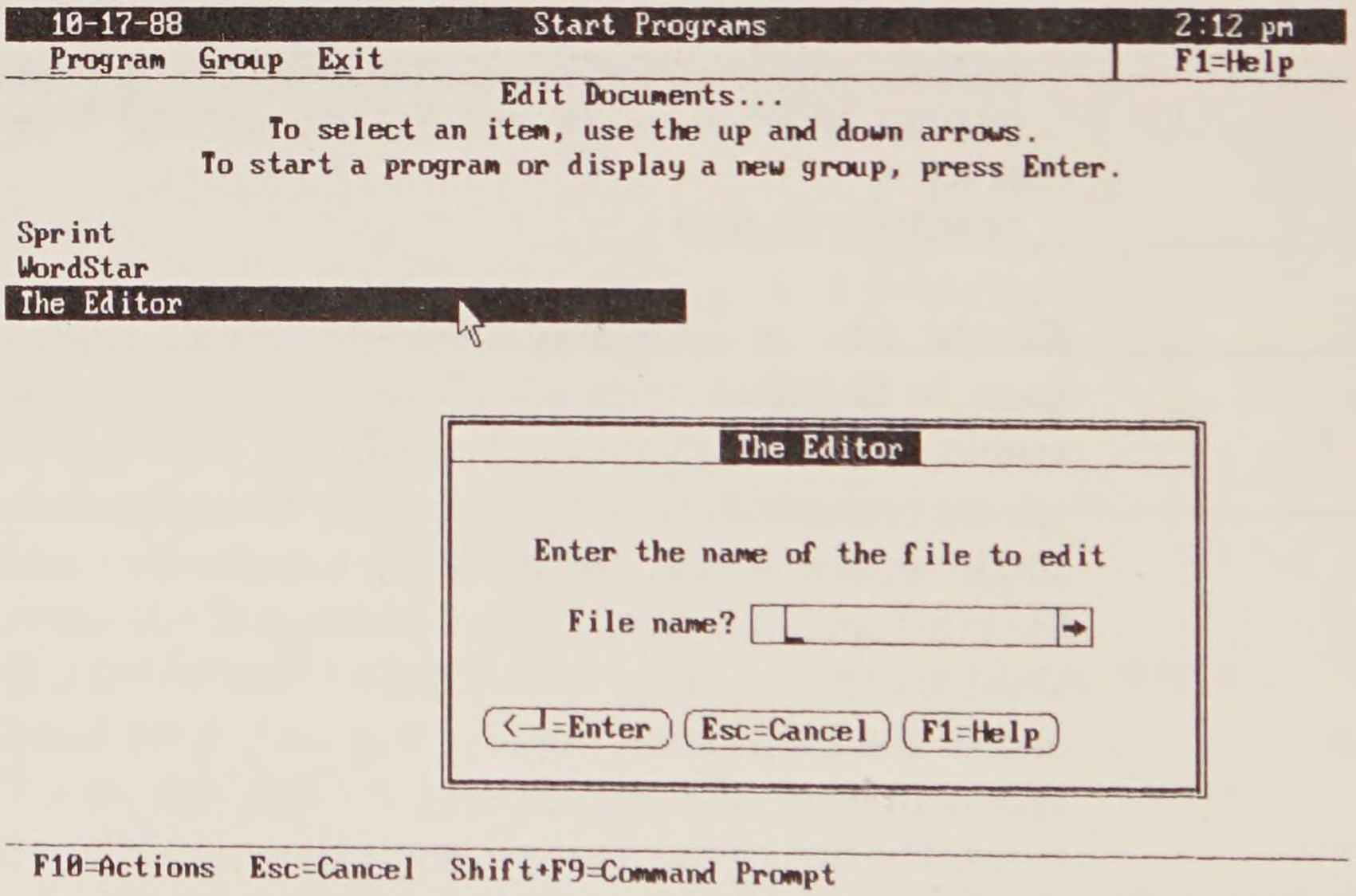


Figure 44: Fully configured program dialog box

point to remember is that DOSShell executes programs by invoking a secondary command processor. Thus, to execute a batch file and return to the shell, the program startup command must begin with CALL. To return to our first example, you want to use the one-line batch file to invoke LOOK.COM, you must enter `[%1 %2 %3]||call look`.

Creating Alternative Shells To use DOSShell to limit access on a multiuser system, create several versions of DOSSHELL.BAT, each with a different version of SHELL.MEU, the file that contains the menu for the Main Group. First create the complete menu system.

You can even install users' names in the Main Group. Or you can require a password both for each individual's group and for the programs within the group. Then issue the commands

```
copy dosshell.bat maintshl.bat
copy shell.meu full.meu
```

and invoke DOSSHELL.BAT, and delete the Command Prompt, Change Colors, and File System from the Main Group. When you have installed only the programs to be accessed from this version of the shell, edit DOSSHELL.BAT and MAINTSHL.BAT.

The line that invokes the memory-resident portion must include a call to the batch file that calls it. I know this doesn't make sense, and I don't know how it works technically, but in MAINTSHL.BAT, you have to change `@shellb dosshell` to read `@shellb maintshl`. If you have any additional versions of the batch file, you must change the shellb parameter in each of them.

In the version that is to provide restricted access, delete the /maint, /exit, /prompt, and /color parameters from the `@shellc` command. The line will then read:

```
@shellc /tran/menu/mul/snd/meu:shell.meu/clr/shell.clr/swap/date
```

The equivalent line in MAINTSHL.BAT would read:

```
@shellc /tran/color/dos/menu/mul/snd/meu:full.meu
  /clr:shell.clr/prompt/maint/exit/swap/date
```

You can easily maintain more than two versions of the shell. Create a separate batch file for each one, following these procedures, with a

different set of programs installed in each version, and a shellb parameter that's the same as the name of the batch file.

Warnings

On one system on which I used the shell, I had no trouble calling the shell from anywhere with the DOS directory appended. But on another—a monochrome system—I kept getting the message `Color file is missing or unreadable` when I invoked the shell from anywhere but its home directory.

There are some bugs in the shell, especially in text mode. In text mode, the shell doesn't reappear properly either after invoking a program from within it, or after using the Command Prompt option. The only way to get the shell back is to execute the batch program again. I haven't encountered this problem in graphics mode.

If you want to load any TSRs that you expect to remove from memory at some point, load them after the first time DOSShell is invoked. If you load them before you invoke DOSShell for the first time, the SHELLB.COM program will be installed in memory above the TSRs you have already loaded, and you won't be able to unload them. (See TSRs.)

The File/Associate command has the unfortunate limitation that few programs have exclusive rights to given file extensions. If several of your programs routinely use the same extensions for various types of files (.DAT, .DOC, and .TXT seem likely candidates) DOSShell won't know which of the programs to start if you select a file with one of these extensions.

Tips

To use different color combinations on different users' menus, first establish the colors you want to use in your maintenance version of DOSShell, then delete the Change Colors option. Next make a copy of the file SHELL.CLR with another name. Include the new file name as part of the /clr parameter in the batch program for the second version of the shell. You can then change the colors on the maintenance version back to your original choice. Repeat the procedure with a different file name for each version of the batch file that calls the shell.

To install prompts or instructions in a Start Program screen that won't fit in the allotted space, install the instruction:

```
[/i"Press F1 for instructions"]
```

and place the instructions on the help line.

If the DOS directory is appended, you can sometimes get away with deleting the first two lines of DOSSHELL.BAT.

DRIVER.SYS

Syntax

{**1**} `device=driver.sys /d:`*n*`/t:`*n*`/s:`*n*`/h:`*nn*`[/n][/c]`
{**2**} `device=driver.sys /d:`*n*`/f:`*n*`[/n][/c]`

DOS Versions

{**all**} MS-DOS 3.2 to 4.0
{**all**} PC-DOS 3.2 to 3.3¦
{**all**} IBM DOS 4.0

Overview

DRIVER.SYS is a device driver you can add to your CONFIG.SYS file to create additional logical drives in your diskette drives or to address external disk drives. The various switches tell DOS the following information:

- The physical location of the drive (/d)
- Whether it's a hard drive (/n)
- The format of the drive (/f; or /t, /s, and /h)
- Whether to let you swap diskettes between two logical drives occupying the same physical drive (/c)

Assigning a new logical drive to an existing physical drive enables you to address that drive by two different drive names. This capability is useful for making some types of copies.

Procedure

Selecting the Appropriate Switches and Values

{**1**} `device=driver.sys /d:`*n*`/t:`*n*`/s:`*n*`/h:`*nn*`[/n][/c]`
{**2**} `device=driver.sys /d:`*n*`/f:`*n*`[/n][/c]`

To install a new logical or physical drive in your system, enter the command `device=driver.sys` in your CONFIG.SYS file, followed by the appropriate switches and the appropriate values in the switches. The switches are as follows:

SWITCH	EFFECT
/d:*n*	Specifies a number associated with a drive letter. Thus, if drive A is 0, drive B will be 1, and so on. For hard disks, see Installing an External Hard Drive, in the next section.
/n	Specifies a hard (nonremovable) disk.
/c	Allows you to swap disks.

and

/t:*n*	Specifies the number of tracks per side.
/s:*n*	Specifies the number of sectors per track.
/h:*n*	Specifies the number of heads. This defaults to 2. You can safely leave it out unless you are installing a single-sided drive or a hard disk.

or

/f:*n*	Specifies the drive type, which Microsoft technicians call the form factor.

PC-DOS, IBM DOS 4.0, and most versions of MS-DOS agree on the following values for /f:*n*:

0 360K, 5¼" diskette drive
1 1.2-megabyte, 5¼" diskette drive
2 720K, 3½" diskette drive
7 1.44-megabyte, 3½" diskette drive, or other

Some versions of MS-DOS use 2 for "other," and some assign a different number to a 1.44-megabyte drive. If you're not sure which number to use, consult your DOS manual; better yet, specify its size in heads, tracks, and sectors. For information on the number of tracks and sectors per diskette type, see Diskettes, Structure of.

The drive name given the new drive depends on what other drives are in your system and where you put the Device= command in your CONFIG.SYS file. Each drive installed with a Device= command automatically receives the next available drive name. Thus, if you have an internal hard disk named C, your new drive will be drive D. However, if you install one or more RAM disks using Device= commands, and these commands precede the command that installs the external drive, the external drive will take the next letter after the RAM disks. If the Device= commands that install RAM disks follow the command that installs the

external drive, the external drive will be drive D. If you do not have an internal hard disk, the device you install first will be drive C.

Installing an External Diskette Drive To install an external diskette drive, include one of the following commands in your CONFIG.SYS file:

```
device=driver.sys /d:n /t:n /s:n /h:nn
```

or

```
device=driver.sys /d:n/f:n
```

If you have two internal diskette drives, the value for /d:*n* will be 2. If you have one, *n* will be 1.

Installing an External Hard Drive

```
{1} device=driver.sys /d:n/t:n/s:n/h:nn/n
```

To install an external hard drive, use the above command filling in the appropriate number of tracks, sectors, and heads. Don't forget the /n switch. The number you assign to /d:*n* depends on the particular version of DOS you use. Some versions require a hexadecimal number of 80 or higher (use 81, for example, if you already have one hard disk in your system). Others require a decimal number of 128 or higher. Still others require a decimal number of 127 or below. Check your DOS manual, the manual for the external hard disk, and, if necessary, call the publisher of your version of DOS if you can't find the information in the manuals.

In any case, the hard drive will have a name not lower than C. It may have some other name, depending on the drives installed before it.

Installing a Second Logical Drive in an Existing Physical Drive To install a new logical drive in an existing physical drive, follow the procedure for installing an external diskette drive. Again, the drive will take the next available drive name. If you want to use this command to copy from one diskette to another of the same type (and there is no other sensible reason for installing a second logical drive in an existing physical drive), add the /c (changeline support) parameter.

Examples

Installing a Second Logical Drive in an Existing Physical Drive It's a good idea to install a second logical drive in an existing physical drive if you have two diskette drives of different types (regardless of whether

one drive is external). The main reason for installing a second logical drive is to simplify copying. With two diskette drives of the same type, you can copy disks and files freely between them.

If you have only one diskette drive, you can normally address it as either drive A or drive B. Thus, you can use the DiskCopy command to copy diskettes from A to B, or the Xcopy or Copy command to copy files (see DiskCopy). However, if you have two diskette drives of different types, you may find it difficult to copy from one diskette to another of the same type. For example, if you have a PC AT with a 1.2-megabyte drive and a 360K drive, you can copy 360K diskettes from one drive to another. You can copy complete 1.2-megabyte diskettes using the command

```
diskcopy a:
```

(See DiskCopy for further information.) If you want to copy some files from one 1.2-megabyte diskette to another, try

```
assign b=a
```

This would give your 1.2-megabyte drive (drive A) the logical names A and B, temporarily disabling drive B. When you enter a command such as

```
copy thisfile.doc b:
```

you won't see any error messages, so you might think the trick has worked. But you also won't get the chance to take the source diskette out and replace it with the target before the command has finished executing. In other words, you will have copied the file onto itself, which is theoretically impossible.

If you do manage to get the source diskette out at just the right moment and replace it with the target, you risk trashing the target diskette completely. DOS keeps a record of the File Allocation Table of the source and the target in memory. If the target diskette doesn't match what DOS thinks it should be, DOS may well write the file to sectors that are already occupied by data (see File Allocation Table).

DRIVER.SYS is the solution to your problem. If you have, for example, a hard disk and two RAM disks in your system, and you install DRIVER.SYS after the RAM disks, drive A will now also be drive G. You can now enter the command

```
xcopy a:*.bas g:
```

Xcopy will then display the message

```
Insert target diskette in drive G:

Press any key when ready . . .
```

DOS will quickly read the FAT, then tell you to reinsert the source diskette. Xcopy will then read as many matching files as memory can hold, then again ask you to insert the target. (Depending on your memory and the amount of data you are copying, you may have to repeat these steps.) Swap disks and you will have copies of the files on another 1.2-megabyte diskette. The only price you pay is the extra 1K or so of RAM that DRIVER.SYS uses.

You can use this same principle for any diskette drive in your system, internal or external, that doesn't have a mate.

Tips

For related information, see CONFIG.SYS File, Device=, and Assign.

Drivparm=

Syntax

{**1**} `drivparm=/d:`*n*`/t:`*n*`/s:`*n*`/h:`*nn*`[/n][/c]`

{**2**} `drivparm=/d:`*n*`/f:`*n*`[/n][/c]`

DOS Versions

{**all**} MS-DOS 3.2 to 4.0

Overview

Drivparm= is a configuration command that you can place in your CONFIG.SYS file to change the description of a physical disk in your system. The various switches give DOS the following information:

- The physical location of the drive (/d)
- Whether it's a hard drive (/n)
- The format of the drive (/f; or /t, /s, and /h)
- Whether to let you swap diskettes between two logical drives occupying the same physical drive (/c)

The syntax and switches are the same as for the DRIVER.SYS device driver (see DRIVER.SYS). The only difference is that each consecutive value for the /d switch represents the next drive letter (that is, 0 is drive A, 1 is drive B, 2 is drive C, and so on), regardless of the type of drive it is. If you use no switches, DOS assumes a 720K, 3½" diskette drive.

Echo

Syntax

{1} echo on
{2} echo off
{3} echo *message*
{4} *@command*

DOS Versions

{1}{2}{3} MS-DOS 2.1 to 3.2
{all} MS-DOS 3.3 to 4.0
{1}{2}{3} PC-DOS 2.0 to 3.2
{all} PC-DOS 3.3¦
{all} IBM DOS 4.0

Overview

Echo is a batch command that controls whether commands in a batch file are displayed while the file is executed. It can also display messages while a batch file is executing.

Procedure

Controlling the Display of Commands

{1} echo on
{2} echo off

Unless you explicitly instruct DOS otherwise, it will display every command in a batch file as it executes the file. To suppress the display of commands, enter the command echo off as the first line of a batch file. This command will appear on the screen, but usually no others will appear.

Turning Echo off does not affect messages generated by the commands. Prompts, warnings, and other output that usually appear on the screen will still appear.

If you have already turned Echo off, other Echo off commands in the same batch file have no effect.

An Echo command with no parameter displays one of the following messages:

```
ECHO is off
ECHO is on
```

You can display commands selectively by turning Echo on during a batch file, and then turning it off again. To turn Echo on, enter the command `echo on` in your batch file.

Suppressing Command Display in DOS 3.3 and Later

{4} *@command*

In DOS 3.3 and later releases, you can suppress the display of any command in a batch file by preceding it with an @ symbol. Thus, the command `@echo off` not only prevents the Echo off command from appearing, but also suppresses the display of all other commands in the file. This allows you precise control of the displays that your batch files generate.

Displaying Your Own Messages

{3} `echo` *message*

When Echo is off, you can still display messages while a batch file executes. To do so, use the Echo command followed by the message. For example, a batch file consisting of the lines

```
echo off
echo Deleting old .BAK files
del *.bak
```

displays the following as it executes:

```
echo off
Deleting old .BAK files
```

You can produce an even cleaner display in DOS 3.3 and later by using the following commands:

```
@echo Deleting old .BAK files
@del *.bak
```

Appending Files

{3} echo *message*

This form of the Echo command can also create a file or add text to the end of a file. To create a file, use the following command to redirect the output to the file:

echo *message >filename.ext*

To append new text to a file, use the append operator, which is two redirection operators:

echo *message>>filename.ext*

This appends the new text to the end of the file. However, if the existing file ends with a Ctrl-Z, you won't see the new text if you display the file with the Type command, since the Type command automatically stops displaying text when it reaches a Ctrl-Z. If your word processor ends files with a Ctrl-Z, you won't be able to see the new text in your word processor either. For examples of this technique of appending files, see Call and Redirection and Pipes.

Examples

Displaying Messages The following program shows a version of DATABAK.BAT (versions of which also appear in the Batch Files entry and in the Rem entry). Note in Figure 80 in the Rem entry how the Rem command displays messages on the screen when Echo is on. The version of the program shown here has Echo turned off, and displays messages using Echo statements rather than Rem statements. If you compare Figures 80 and 45, you will see that Echo statements, judiciously used, create a more readable display than do Rem statements.

DATABAK.BAT, with Messages in Echo Statements

```
echo off
c:
cd\dos
echo Any data on the diskette in drive B will be lost
format b:
echo Verifying the integrity of the diskette in drive B.
echo If you see a message regarding bad sectors,
echo press Ctrl-C and start over.
chkdsk b:
echo Copying files from drive A to drive B
xcopy a:. b: /v
dir b:
```

You can usually make your batch files more elegant by turning Echo off at the beginning, then using the Echo and Type commands to display screens or messages. However, sometimes it's useful to turn Echo on during a batch file so that you can see what the batch file is doing. UPDATE.BAT, shown in the entry on Hard Disks, Backing Up and Restoring, is an example of this technique.

Tips

Turning Off Messages from Programs To turn off both messages generated by commands that appear in batch files and the display of the commands themselves, end the command line with

```
>nul
```

This redirects messages to the null device (often called the bit bucket), which is to say, nowhere. This technique does not suppress all messages, however. For example, if you attempt to delete a file that isn't there, you will still see a `File not found` message. Also, some programs override the `>nul` comand and display messages whether you want them to or not. See Ctty for a method of suppressing messages altogether.

Blank Screen Lines in Batch Files To create a blank line on the screen when Echo is off, enter a statement of the form

```
echo ^@
```

```
D:\>databak


D:\>echo off
Any data on the diskette in drive B will be lost
Insert new diskette for drive B:
and strike ENTER when ready

Format complete

    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/N)?n
Verifying the integrity of the diskette in drive B.
If you see a message regarding bad sectors,
press Ctrl-C and start over.

   362496 bytes total disk space
   362496 bytes available on disk

   655360 bytes total memory
   268416 bytes free
Copying files from drive A to drive B
Reading source file(s)...

A:TOTALS.DAT
A:ADDRESS.MCD
A:MAX_FIGS.FW2
A:CA611.LTR
A:CA910.LTR
A:CULLINAN.LTR
A:MCD1013.LTR
A:SUBMIT.1
        8 File(s) copied

 Volume in drive B has no label
 Directory of  B:\

TOTALS   DAT      256  10-13-87   2:14p
ADDRESS  MCD      256  10-13-87   2:15p
MAX_FIGS FW2     7696   9-14-87  10:52a
CA611    LTR      768   6-11-87   1:05p
CA910    LTR     1920   9-10-87   2:15p
CULLINAN LTR     3072   9-22-87   2:16p
MCD1013  LTR     3072  10-13-87   2:04p
SUBMIT   1       1476  10-13-87   2:44p
        8 File(s)    340992 bytes free
```

Figure 45: Output from DATABAK.BAT with Echo statements

The ^@ character is a null (ASCII 0). You can enter it in EDLIN by typing *Ctrl-V* @. If you're entering the file using the Copy CON command, you can enter a null by pressing F7 (see Copy CON). If you use a word processor that cannot enter a null (as many cannot), you can substitute an innocuous character such as a period.

Turning Off the Prompt Whether or not you turn Echo off in a batch file, Echo is always on after a batch file executes. However, typing `echo off` at a DOS prompt hides the DOS prompt, just as it does in a batch file. On the other hand, the screen and keyboard are still active, and you can enter commands as usual. If you have entered such a command, the prompt will not return after a batch file unless the batch file contains an Echo on statement.

Debugging Batch Files Avoid turning Echo off when debugging batch files. Otherwise, you won't know which command is generating any error message that may appear.

For related information, see Batch Files, Pause, Rem, and Type.

EDLIN

Syntax EDLIN is a DOS utility program with many commands of its own. Since each procedure in EDLIN uses different commands, the syntax is listed separately for each operation.

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

EDLIN is a quick-and-dirty line editor supplied free with all versions of DOS. As editors go, it's user-hostile and counterintuitive. EDLIN is not much better than the Copy CON command for short, unformatted text files, and it's much worse than almost any word processor for producing formatted text. However, EDLIN is adequate for minor editing tasks such as changing a few characters in a batch file you created with Copy CON, or inserting a Ctrl-Z at just the right point in a display screen.

Line Logic Being a line editor, EDLIN lets you edit one line of your file at a time. It regards a line as a logical unit having a maximum of 253

characters. Each line has a number, based on its position in the file. All subsequent lines are renumbered if you insert or delete lines in a file.

EDLIN wraps lines at the right edge of the screen (even in the middle of a word). Once the cursor has moved to the next screen line, you cannot move it up to the previous one, even if you are still on the same logical line.

The DOS editing and command keys all work in EDLIN as they do at the DOS prompt.

Memory Limitations If a file is larger than available memory, EDLIN loads text into memory until it is three-quarters full. It then displays its prompt, an asterisk (*), and you can add text to the file until memory is full. At that point, you must write some of the file to disk and read in some of the remainder, as explained later in this entry. If the entire file takes up less than three-quarters of available memory, EDLIN reads the entire file into memory and displays the message `End of input file`. A maximum of 65,529 lines is allowed in a file.

General Rules for EDLIN Commands All EDLIN commands are entered by typing their first letter. For example, Insert is *i*, Delete is *d*, Move is *m*, and so on. All commands in EDLIN apply to either a specific line or a group of lines. You specify lines by number or by using one of several special characters. To specify a range of lines, enter two line numbers separated by a comma. The line numbers precede the letter representing the command. You may use spaces before or after the commas and before the command character. You can also omit them; it makes no difference.

If you do not specify a line number, the current line—that is, the line most recently edited—is used. You can specify any single line by number. You can specify a group of lines either by number or by a range surrounding the current line. For example, to display (list) lines 3 through 12 of your file, you enter the command `3,12l`. If your current line is line 10, you can produce the same effect by entering the command `-7,+2l`. This command asks EDLIN to list the seven lines before the current line and the two lines after it. You can use either form with any command that acts on a range of lines.

When using either absolute or relative line numbers, you may specify only one end of the range. Depending on the command, EDLIN will either assume that the other end is the current line or act on twenty-three lines preceding or following the specified line.

The EDLIN Screen When you first enter EDLIN, an asterisk appears at the left margin. This asterisk, which is EDLIN's command prompt, is also EDLIN's way of telling you which line is current. When you begin working, your text appears on numbered lines, indented several spaces from the left edge of the screen. (The number of spaces varies in different releases of DOS.) Each line has a number followed by a colon. The current line shows the asterisk immediately following the colon. On other lines, a space separates text from the colon.

Procedure EDLIN is a DOS external program and using it requires having EDLIN.COM (or EDLIN.EXE) on a disk in your computer. If you use DOS 2, EDLIN must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface EDLIN with a drive name; for example,

```
b:edlin newfile
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\edlin a:config.sys
```

Creating a New File

`edlin [drive][path\]filename.ext`

To create a new file, invoke EDLIN with the name of the file you want to create as its parameter. You may specify a drive and/or path name for the file if you want it to be on a directory other than the current one:

```
edlin \batch\test.bat
```

You will see the message

```
New file
*
```

The asterisk, EDLIN's prompt, tells you that you are in EDLIN and that EDLIN is ready to accept a command.

Editing an Existing File

`edlin [drive][path\]filename.ext [/b]`

The syntax for invoking EDLIN to edit an existing file is the same as that for editing a new file. When you edit an existing file, EDLIN makes a copy of the file with the extension .BAK before opening it. Thus, you still have a copy of the file as it existed before editing. EDLIN then reads as much of the file as it can into memory.

To edit a file that may include Ctrl-Z characters, use the /b (binary) switch in your command line. This switch tells EDLIN to read the file to its end as recorded by DOS. Otherwise, it will stop reading at the first Ctrl-Z it finds.

Inserting Lines

{1} `[n]i`
{2} `#i`

To create a new file, you must tell EDLIN you want to *insert* lines. With a new file, the command `i` places you on the first line. Lines are set off by a line number and a colon, and an asterisk following the colon marks the current line. With a new file, the Insert command produces this result:

```
1:*_
```

You can enter up to 253 characters following the line number. If you press Return, you will be given a new line:

```
1:*This is the first line of my new file.
2:*_
```

Pressing Return on a blank line creates a blank line:

```
1:*This is the first line of my new file.
2:*
3:*This is the third line of my new file.
4:*_
```

To do something other than insert lines, you must leave Insert mode either by pressing Ctrl-C, or by pressing Ctrl-Z and pressing Return, as Figure 46 shows.

To enter a new line at a point other than the current line, return to the EDLIN prompt and enter the number of the line to insert:

```
2i
```

This creates a new line 2, thereby changing the old line 2 into line 3:

```
2:*I forgot to enter text on the second line.
```

All higher line numbers will increase as well. If you continue inserting new lines, succeeding lines will be renumbered accordingly.

{2} #i

To insert new text at the end of the file, precede the Insert command with an octothorp, the number/pound symbol:

```
#i
```

This tells EDLIN to make the line after the last line current.

Viewing Your File

{1} [*n*][,*n*]l
{2} [-*n*][,+*n*]l
{3} .l
{4} [*n*][,*n*]p
{5} [-*n*][,+*n*]p

You can view your file using either the List or the Page command.

{1} [*n*][,*n*]l

```
1:*This is the first line of my new file.
2:*
3:*This is the third line of my new file.
4:*^C
*_
```

Figure 46: Leaving EDLIN's Insert mode

The List command displays up to twenty-three lines at a time, centering on the current line, unless you ask for specific lines. To list lines preceding the current line, enter the current line followed by a comma (think of it as the first of two parameters):

```
2,l
```

The screen would display up to twenty-three lines following line 2. Similarly, ,26l would display twenty-three lines ending in line 26.

If your logical lines are more than one screen line long, you will want to restrict the range so that part of the range doesn't scroll off the screen:

```
12,26l
```

{2} [-*n*][,+*n*]l

Like all EDLIN commands that refer to line ranges, List can accept relative line numbers. Thus, if line 20 is current, the command -8,+6l has the same effect as 12,26l.

{3} .l

To list lines beginning with the current line, enter a period as the parameter:

```
.l
```

If you enter this command with the current file, you see

```
*.l
3:*
4: This is the third line of my new file.
*_
```

{4} [*n*][,*n*]p

{5} [-*n*][,+*n*]p

The Page command displays twenty-three lines at a time beginning with the current line; the last line displayed becomes the current line. Thus, if you enter the Page command repeatedly, EDLIN scrolls through your file until the end, leaving the last line current. You can specify a starting line; for example, 12p will display twenty-three lines beginning with line 12, leaving line 35 current. Specifying a starting and ending line or a relative

range works the same as it does using the List command. The only difference is that the last line displayed becomes current. Note that you can use relative as well as absolute line numbers.

Deleting Lines

{1} [*n*][,*n*]d
{2} [-*n*][,+*n*]d

The Delete command deletes one or more lines. If you enter the d command with no parameters, it deletes the current line. If you enter a line number before the d, it deletes the specified line. If you enter two line numbers separated by a comma, EDLIN deletes both specified lines and all lines in between.

As with the List command, you can enter a single parameter, preceded or followed by a comma. EDLIN assumes that the missing parameter is the current line. If line 2 is current, the command ,26d deletes lines 2 through 26. The former line 27 becomes the new line 2, and all following lines are renumbered accordingly.

To erase the blank third line in the sample file, enter

```
3d
```

If you then list the file, the screen will look like Figure 47.

You can delete lines using relative as well as absolute numbers. Using relative numbers is a bit scarier than absolute numbers because it's easier to make a mistake. Before deleting lines, look at the portion of the file containing the lines you want to delete to make sure you have the line numbers right.

There is no way to undelete lines once you have deleted them—they are gone forever. If you are editing an existing file, you can recover the

```
*3d
*l
        1: This is the first line of my new file.
        2: I forgot to enter text on the second line.
        3:*This is the third line of my new file.
*_
```

Figure 47: Deleting in EDLIN

deleted lines by quitting the EDLIN session and starting over, as explained below. However, if you are editing a new file, there is no way to get the deleted lines back.

Editing Existing Lines To edit a line in your file, enter the number of the line you want to edit. EDLIN displays the line and presents you with a blank line of the same number. To edit line 2 of the sample file, enter

```
2
```

The screen will display

```
*2
2:*I forgot to enter text on the second line.
2:*_
```

You can then either enter a completely new line on the second line or edit the text that is already on it. You use the DOS editing keys to edit (see Command Keys). As in DOS, the current line becomes an editing template. Use the right-arrow or F1 key to display a character at a time from the template, or the F3 key to display the entire line. The generally useless F2 and F4 keys work the same way in EDLIN as they do at a DOS prompt.

To delete characters, press the Del key once for each character you want to delete. To insert characters at a given point, first bring the characters from the template to the editing line with the editing keys. When you reach the point where you want to insert characters, press the Ins key and enter the characters. When you finish inserting, the template keys will bring out the remainder of the line if you wish to use it.

If you enter characters without pressing Ins first, they will replace the characters in the same position in the template.

As soon as you press Return, the text of the current line is saved to the template, and the old template disappears.

To edit the second line of our file, Press F1 twice. You now see

```
*2
2:*I forgot to enter text on the second line.
2:*I _
```

Now type

```
have now
```

These characters will replace *forgot t*. Press Del to delete the *o*. Hold down the F1 or the right-arrow key until the word *enter* (but not the space following it) appears on the line. If you go too far, the backspace or left-arrow keys will remove the extra characters. Press Ins and type

```
ed
```

Next, press F3 to bring back the rest of your line, and press Return. The screen will show

```
*2
2:*I forgot to enter text on the second line.
2:*I have now entered text on the second line.
*_
```

and the original text of line 2 will be gone.

Copying Lines

{1} [*n*],[*n*],*t*[,*c*]c
{2} [-*n*],[+*n*],*t*[,*c*]c

To copy one line from one part of a file to another, you must specify the target of the copy as a line number. The lines you copy will appear before the target line. If you specify only the target, EDLIN copies the current line to the line before the target line. If you specify a range, EDLIN copies the entire range. If you specify a range that includes the target, EDLIN displays the message `Entry error` because it cannot do what you requested.

You can make more than one copy at a time by specifying the number of copies. You must be careful, however. If you enter three parameters, EDLIN assumes that they are a starting line, an ending line, and a target line. If you want to make several copies of a single line, specify that line as both the starting and ending line:

```
1,1,4,6c
```

This command makes six copies of line 1, placing them before line 4 (or after line 3, if you prefer to think of it that way).

If the target line is more than one line beyond the end of the existing file, and you copy enough lines to create such a line, the target line becomes current after the copy. If the target line number you specified is too high, the lines will be copied to the end of the file and no line will be current.

As with all the other commands, you can specify a range to copy with numbers relative to the current line. You can also specify one end of the range and let EDLIN assume that the other end is the current line.

Moving Lines

{1} [*n*],[*n*],*t*m

{2} [-*n*],[+*n*],*t*m

To move one or more lines from one part of a file to another, you must specify the target line before which the moved lines should appear. If you do not specify the lines to move, EDLIN moves only the current line. If you want to move a single line that is not current, specify it twice. Thus, to move line 2 to where line 4 was, enter

```
2,2,5m
```

In other words, move line 2 to before line 5. This process is more confusing than it seems because all the lines following the original location of the moved line will be renumbered.

Let's add two lines to your file and see what happens:

```
4: If I keep this up, my file might eventually get long.
5: It now has five lines.
```

To do so, type 4i and enter the text, pressing Return at the end of the line. Press Ctrl-C when you are finished. Now enter

```
2,2,5m
```

When you list the file, you will see what appears in Figure 48.

```
        4:*If I keep this up, my file might eventually get long.
        5:*It now has five lines.
        6:*^C
 *2,2,5m
*l
        1: This is the first line of my new file.
        2: This is the third line of my new file.
        3: If I keep this up, my file might eventually get long.
        4:*I have now entered text on the second line.
        5: It now has five lines.
*_
```

Figure 48: Moving lines in EDLIN

Searching for Text

[*n*][,*n*][?]s*text*

EDLIN's search command is rather persnickety about where it's willing to search for text. If you omit the parameter specifying the beginning of the range to search, EDLIN starts after the current line and goes to the end. If you don't specify the end of the range, it searches from the specified line to one line past the last line. If you omit both ends of the range, it searches from the line after the current line to the line after the end. Omitting both ends of the range is the same as entering

```
.,#stext
```

Remember that the period stands for the current line, and the octothorp (number/pound symbol) for the line after the last.

After you have specified the range to search, enter the s command, followed by the text you want to search for. If EDLIN finds the text, it will make the line containing it current. Otherwise it displays the message `Not found`.

Let's try this with the current file. By now you may have lost track of your original second line. If you enter `ssecond`, you'll see the `Not found` message because line 4, which contains the text, is current, and the search starts after the current line. Let's try searching the whole file:

```
1,#ssecond
```

This time, the correct line appears:

```
*1,#ssecond
4:*I have now entered text on the second line
*_
```

If there is perhaps more than one instance of the search text in the file, use the ? modifier. EDLIN will display the prompt

```
O.K.?_
```

If you press Y, the search will stop. If you press any other key, the search will continue, stopping at each instance of the search text until you press Y, or you reach the end of the file. Let's use the word *line* as an example. The result appears in Figure 49.

```
*1,#?sline
        1: This is the first line of my new file.
O.K.? n
        2: This is the third line of my new file.
O.K.? n
        4:*I have now entered text on the second line.
O.K.? n
        5: It now has five lines.
O.K.? n
Not found
*_
```

Figure 49: Searching for text in EDLIN

Replacing Text

[*n*][,*n*][?]r[*text1*][^Z*text2*]

EDLIN's Replace command follows the same rules as the Search command regarding where it will search if you omit line parameters. (See the previous section for a discussion.)

You must separate the original text from the new text with a Ctrl-Z. If you omit the search text parameter (*text1*), EDLIN will use the last search text you entered. If you leave out the replacement text parameter (*text2*), it will delete the search text.

Generally, the Replace command works much like the Search command. If you do not use the ? modifier, it will replace all instances of *text1* with *text2* without asking your permission. If you do include the ? modifier, EDLIN will stop at each instance, display the line with the altered text, and prompt you with

```
O.K.?
```

If you press Y or Return, EDLIN will replace the text. If you press any other key, it will not.

Let's try this with our file. We'll replace *file* with *document*, getting permission. Enter the command

```
1,#?rfile^Zdocument
```

The result is shown in Figure 50. Try replacing automatically by replacing *new* with *sample*:

```
1,#rnew^Zsample
```

EDLIN displays only the lines in which it replaces text, as Figure 51 shows.

```
*1,#?rfile^Zdocument
        1: This is the first line of my new document.
O.K.? y
        2: This is the third line of my new document.
O.K.? y
        3: If I keep this up, my document might eventually get long.
O.K.? y
*l
        1: This is the first line of my new document.
        2: This is the third line of my new document.
        3:*If I keep this up, my document might eventually get long.
        4: I have now entered text on the second line.
        5: It now has five lines.
*_
```

Figure 50: Queried replacement of text in EDLIN

```
*1,#rnew^Zsample
        1: This is the first line of my sample document.
        2: This is the third line of my sample document.
*_
```

Figure 51: Automatic replacement of text in EDLIN

Adding Text from Another File

`[t]tfilename.ext`

To insert text from another file into your current file, use the Transfer command. If you do not specify a target line, the text will be added before the current line, and the current line and all subsequent lines will be renumbered. You can specify a drive and/or path name from which to read the file if it is not in the current directory.

As you know, EDLIN can handle lines up to 253 characters long. If the file you add to your file has longer lines, EDLIN will truncate them, as Figure 52 shows.

Writing Text from Memory to Disk

`[n]w`

If your file is too large to fit in memory, you may eventually want to bring the rest of the file into memory to edit it. This requires taking

```
*t3longline.txt
*l
        1: This is the first line of my sample document.
        2: This is the third line of my sample document.
        3: This is a text file that is designed to demonstrate the limitations o
f EDLIN when dealing with extremely long lines. As you know, EDLIN can deal with
 only up to 253 characters per line. In this particular peculiar piece of prose
the text is constructed i
        4: I have now entered text on the second line.
        5:*It now has five lines.
```

Figure 52: Inserting text from another file in EDLIN

something out of memory to make room. Use the Write command to do so. With no parameters, the Write command writes text from the beginning of the edit buffer to disk, until memory is only one-quarter full. If you enter a line parameter, it writes only the number of lines you specify.

In either case, the remaining lines will be renumbered so that the portion of the file in memory begins with line 1.

You can now bring some of the file that was left on disk into memory with the Append command, described below.

Bringing Text from Disk to Memory

[*n*]a

If your entire file was not loaded into memory, the Append command brings the remainder into memory once you make room for it with the Write command, described above. If you enter it with no parameters, EDLIN adds the remainder of the file to the end of the part currently in memory and displays the message

```
End of input file
```

If you specify a number of lines, only that number of lines will be added to the file from disk.

Ending the Session

{1} e
{2} q

When you finish editing, EDLIN gives you the choice of saving or abandoning your work. To save your file, use the End command. At an asterisk prompt type e and press Return. EDLIN will save your file and return you to the DOS prompt.

If you decide your original file was better than the edited version, or you've made a hopeless mess, use the Quit command to quit without saving your file. (If you are editing a new file, there will be no previous version to return to.)

At an asterisk prompt, type q and press Return. EDLIN will ask you

```
Abort edit (Y/N)_
```

If you press any key but Y, you will be back in EDLIN, exactly where you left off. If you press Y, EDLIN will rename the original file (to which it gave a .BAK extension when you began editing) to its original name. Then it will return you to the DOS prompt, leaving all your work suspended in electronic memory.

Tips

Control Characters and EDLIN The way EDLIN handles control characters is something of a double-edged sword. On the one hand, you can enter all of them; on the other hand, they won't necessarily stay put. As a general rule, those control-key combinations that act as command keys in DOS do the same in EDLIN (see Command Keys). However, you can enter them in a file if you first enter a Ctrl-V, and then enter the uppercase equivalent of the control character. Thus, to enter a Ctrl-C, you would type `Ctrl-V C`.

Characters such as tabs, line feeds, backspaces, and carriage returns behave the same way in EDLIN as they do in a file you display with the Type command (see Type). See Figure 53 for ways of entering control characters and of the effects of these control characters.

On the other hand, you can enter each one of the high order graphics characters in EDLIN using the Alt key and the numeric keypad (see ASCII Code).

```
        8:*^V@^VC^VF^VH^VI^VJ^VM^VU^VV^VW
        9:*Notice that the Tab (Ctrl-H), line feed (Ctrl-J), and carriage return
 (Ctrl-M) function the same way if you enter them with Ctrl-V as they would if y
ou entered them directly.
       10:*This is an Escape character: ^V[
       11:*Ctrl-Z has the same effect as Ctrl-C. It brings back the prompt.
       12*^C
*l
        1: This is how EDLIN deals with control characters.
        2: Entering Ctrl-Shift-2 (Ctrl-@) issues a Ctrl-C, returning you to the
prompt.
        3: Any control character that you can enter at a DOS prompt can be enter
ed in EDLIN the same way, except Ctrl-V.
        4: ^A^B^D^E^G^K^L^N^O^Q^R^T^X^Y^\^]^_ ^^
        5: The control keys that function as editing keys in DOS do so in EDLIN
as well.
        6: Ctrl-C returns you to the EDLIN prompt instead of the DOS prompt; Ctr
l-H is a Tab; Ctrl-I is a Backspace; Ctrl-J is a line fee"; Ctrl-M is a carriage
 return; Ctrl-U erases to the beginning of the line; Ctrl-W erases the previous
word.
        7: Use Ctrl-V before an uppercase character to enter the character as a
control character. This is how you enter the characters t"at you cannot enter di
rectly.
        8: ^@^C^F^H
^U^V^W                          9: Notice that the Tab (Ctrl-H), line feed (Ctrl-
J), and carriage return (Ctrl-M) function the same way if you enter them with Ct
rl-V as they would if you entered them directly.
       10: This is an Escape character: ^[
       11: Ctrl-Z has the same effect as Ctrl-C. It brings back the prompt.
*_
```

Figure 53: Control characters in EDLIN

Environment Space

Overview

The environment space is a section of memory associated with the command processor (see Command). An environment is created automatically whenever a program is loaded. It contains the following data:

- The name and location of the current command processor
- The current search path (see Path)
- The form of the prompt if you have established one (see Prompt)
- Your list of appended directories, if you loaded Append with the /e switch (see Append)
- Any variables established using the Set command (see Set)

These items are stored as a series of strings terminated with a null (zero byte). Such strings are also called ASCIIZ strings. The entire environment space ends with a null.

The default size of the environment is 160 bytes. If you do not load any TSRs, DOS can expand the environment in 127-byte increments as needed. However, once you have loaded a TSR, the environment space cannot be expanded further (see TSRs).

Whenever you load a program, the command processor creates an environment for it (see Command). The new environment will "inherit" any values in the previous command processor's environment. However, if you make any changes, such as adding environment variables or changing the search path, these changes will disappear when you exit from the secondary command processor because its environment will disappear along with it (see Exit).

Procedure

Environment Variables To view the current values assigned to environment variables, use the Set command (see Set). At minimum, the environment will always contain the default path and the location of the current command processor. The latter is revealed in a statement of the form

```
COMSPEC=drive\[path\]FILENAME.EXT
```

where *drive* is the boot drive, and *FILENAME.EXT is the name of the command processor. Unless you use some unusual applications, this will be COMMAND.COM. You can change the default command processor in DOS 3 by using a command of the form*

```
set comspec=drive[path\]filename.ext
```

If you have not issued a path statement, the environment will also show

```
PATH=
```

If you have issued a path statement, the specified path will be shown.

Enlarging the Environment If you use a long search path or append list, a complex prompt, or many environment variables, you may see the message

```
Out of environment space
```

If you see this message, some of the values you have attempted to place in the environment may be missing or truncated. For example, if your path statement was the last item placed in the environment before you saw this message, the last few directories on the search path may be missing.

You can expand the environment in DOS 3.1 and later using the Shell= or Command commands to anywhere from 160 to 32,768 bytes, in 16-byte increments (see Shell= and Command). The default environment is adequate for most users. However, I have occasionally needed an

environment as large as 1088 bytes when using a path of maximum length, a complex prompt that included several ANSI escape sequences, and several programs that placed strings in the environment.

Tips

If you program in C, it is relatively easy to read environment variables with the GETENV function.

For related information, see Command, Set, and Shell=.

Erase. *See* Del

ErrorLevel

Syntax

```
if [not] errorlevel n command
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

ErrorLevel is a modifier to the If batch command (see If). ErrorLevel tests for *exit codes*. An exit code is a numeric value generally used to indicate whether the program worked properly or, if it didn't, what went wrong.

The following DOS external commands generate exit codes:

Backup
DiskComp
DiskCopy
Format
GrafTabl
Keyb (but not Keyb*xx*)
Replace
Restore
Xcopy

The code 0 indicates that the command was successfully executed. Any other code indicates an error condition of some sort.

In any given manufacturer's version of MS-DOS, these commands may or may not generate exit codes, and other commands may generate such codes as well. This list is based on PC-DOS 3.3 and generic MS-DOS 3.2. The codes appear in Table 14.

Procedure

The ErrorLevel modifier can be used only in a batch file. Include in the batch file a command that generates an ErrorLevel code, then follow it with a number of If statements directing that specific actions be taken depending on the exit code.

ErrorLevel responds to values greater than or equal to the one you specify. Therefore, you normally test for the highest possible exit code first and construct your batch file so that the lines testing for lower exit codes are not executed if a higher code is valid. Statements including the Not modifier and nested If statements provide two ways of streamlining the process, as the examples that follow demonstrate.

TABLE 14

Exit Codes for DOS External Programs

CODE	MEANING
Backup Program	
4	Program ended because of an error.
3	Program interrupted by Ctrl-Break.
2	Some files were not backed up because of file-sharing conflicts.
1	There were no files to be backed up.
0	No errors occurred in executing the command.
DiskComp Program (MS-DOS only)	
4	There was not enough memory, an invalid drive was specified, or there was a syntax error on the command line.
3	Program was not executed because of a read-write error.
2	Program interrupted by Ctrl-Break.
1	The disks were not the same.
0	The disks were identical.

Table 14 continues

CODE	MEANING
DiskCopy Program (MS-DOS only)	
4	There was not enough memory, an invalid drive was specified, or there was a syntax error on the command line.
3	The source disk was unreadable or the target disk could not be formatted.
2	Program interrupted by Ctrl-Break.
1	A nonfatal read-write error occurred.
0	The diskette was copied successfully.
Format Program	
5	You specified a hard disk, then pressed N when asked whether to proceed.
4	Fatal error other than those specified.
3	Program interrupted by Ctrl-Break.
0	The target disk was formatted successfully.
GrafTabl Program (PC-DOS 3.3, MS-DOS 3.2 and later)	
4	Your version of DOS does not match that of the program.
3	You entered an incorrect parameter; the program was not executed.
2	No character table was loaded.
1	A new character table replaced a previously loaded character table.
0	The character table that you specifed was successfully loaded.
Keyb Program (DOS 3.3 and later)	
6	The translation table for the character set (code page) you specified was not in the resident keyboard table (KEYBOARD.SYS).
5	You didn't prepare a code page request with the Mode command; therefore, the keyboard definition cannot be loaded.
4	A screen or keyboard error occurred.
3	The keyboard table could not be successfully placed in memory.
2	The keyboard definition file is bad or missing.
1	Your command included a syntax error, a request for a nonexistent code page, or a request for a language not supported by the Keyb driver.
0	The keyboard table you requested was successfully placed into memory.

Table 14 continues

CODE	MEANING
Replace Program[1]	
15	You specified an invalid drive.
8	There was not enough memory available to execute the command.
5	The target was marked read-only, and therefore not replaced.
3	The specified path was not found.
2	No matching files were found.
1	You made an error on the command line.
0	All matching files were successfully replaced.
Restore Program	
4	Program ended because of an error.
3	You interrupted the program with a Break.
2	Some files were not restored because of file-sharing conflicts.
1	There were no files to be restored.
0	No errors occurred in executing the command.
Xcopy Program	
5	An Interrupt 24 error occurred.
4	There was either an error in the command line or not enough memory to execute the command pattern on the source directory.
0	No errors occurred in executing the command.

[1]**Note:** Replace also uses all the standard DOS error codes.

Examples

CHKCOPY.BAT, which follows, responds to the exit codes for the Xcopy command. (Exit codes for all the DOS external programs that have them appear in Table 14.) Notice that Xcopy does not have a code 3. CHKCOPY.BAT tests for the exit codes and displays an appropriate message depending on the code generated by Xcopy.

CHKCOPY.BAT

```
echo off
xcopy %1 d:
if errorlevel 4 echo ERROR! Reboot computer and start over.
if errorlevel 4 goto end
if errorlevel 2 echo BREAK issued. Copy interrupted.
if errorlevel 2 goto end
if errorlevel 1 echo No such file(s) on source directory.
if not errorlevel 1 echo Copy successful.
:end
```

There is no test for code 5; the test for code 4 also responds to code 5 because the ErrorLevel modifier responds to codes greater than or equal to the one specified in the batch file statement that uses it. Thus, the message applies to both conditions. It may be a bit extreme to tell the user to reboot when the problem might only be an invalid drive in the command line argument; however, this is just an illustration.

If Xcopy generates a 5 or 4 exit code, the tests for 2, 1, and 0 will also respond to it. Therefore, you must use a GoTo statement to direct execution past these tests.

Similarly, if you create exit code 2 by pressing Ctrl-Break, the tests for codes 1 and 0 will respond; therefore, another GoTo statement is necessary.

You can save a line of code by using the syntax

```
if not errorlevel 1
```

instead of

```
if errorlevel 0
```

Because the test for code 1 responds to all higher codes, the test for the absence of code 1 ignores all higher codes. The alternative would be

```
if errorlevel 1 goto end
if errorlevel 0 echo Copy successful.
```

The *script file* that follows is used as input to DEBUG (see Redirection and Pipes). This file creates a program called FRIDAY.COM.

To create this program, you must first create the file as an ASCII file, using a word processor in nondocument mode or EDLIN (see EDLIN). Be sure to enter a carriage return on the blank line and after the last line. If you forget to do so, the computer will lock up when you try to create the .COM file. The next step is to be sure that DEBUG is in the current directory or on the path. Then enter the command `debug<friday.deb`.

FRIDAY.DEB

```
n friday.com
a
mov ah,2a
int 21
cmp al,05
jnz 0113
mov al,00
cmp dl,15
jnb 115
mov al,01
jmp 0115
mov al,02
mov ah,4c
int 21

rcx
19
w
q
```

FRIDAY.COM indicates whether today is the last Friday of the month (assuming you have set the date correctly). This is used in the HDBACKUP.BAT program in the entry Hard Disks, Backing Up and Restoring, to determine whether it is time to do a complete backup. FRIDAY.COM works by generating one of three exit codes: If it's the last Friday of the month, exit code 0 is generated; if it's Friday, but not the last, exit code 1 is generated; if it's not Friday, exit code 2 is generated.

Since the program produces an exit code, the only way to retrieve the result is to test for it with If ErrorLevel statements. TESTFRI.BAT, shown below, performs the necessary tests.

TESTFRI.BAT

```
echo off
friday
if errorlevel 0 if not errorlevel 1 echo Last Friday of the month.
if errorlevel 1 if not errorlevel 2 echo It's Friday.
if errorlevel 2 echo It's not Friday.
```

Two aspects of TESTFRI.BAT are particularly noteworthy. First, note the nested If statements on each line. The syntax

```
if errorlevel n if not errorlevel n+1 . . .
```

tests for the lower code and ignores any higher codes. This decreases by one-half the number of lines of code because you don't need a GoTo statement after each test.

Second, you don't need the second test if all you need to know is whether it's the last Friday. This allows you to reduce the program by a line to take advantage of the fact that any code greater than the tested code also activates the response. Thus, all you really need is the version of the program shown below.

TESTFRI.BAT, Version 2

```
echo off
friday
if errorlevel 0 if not errorlevel 1 echo Last Friday of the month.
if errorlevel 1 echo It's not the last Friday of the month.
```

Tips

If you program in a high-level language, you can have a program produce an exit code by moving the appropriate value into the AL register before terminating. See FRIDAY.DEB in this entry for an example.

For another program using ErrorLevel to test exit codes, see DATEBAK.BAT in the Xcopy entry.

EXE2BIN

Syntax

{1} exe2bin *filename*
{2} exe2bin *filename filename*.com

DOS Versions

{all} MS-DOS 2.1 to 4.0
{all} PC-DOS 1.1 to 3.2

Overview

As its name implies, EXE2BIN converts some executable files (with an .EXE extension) to binary, or command, files (with a .BIN extension). If you include the .COM extension in the target file name, the result will be a .COM file.

DOS allows for two different kinds of executable files. Although of little significance to the average user, these differences are important both to DOS and to the programmer.

Command (.COM) files must be less than 64K in length. Their program code loads directly into memory, and represents a sort of snapshot of the code in memory required to perform the program's tasks.

A program longer than 64K must use more than one memory segment (see Memory). The registers in which segment addresses are stored cannot hold addresses longer than 64K. Because a program that is created by linking several segments cannot determine the location of the segments at time of loading, a linked program has all references to segment locations in its code filled with zeros.

The header of an executable file (one with an .EXE extension) contains a list of all references to segments. When DOS loads the file, it looks in the header and changes the zeros to the correct addresses; in other words, it supplies the segment references that could not be known at link time. This type of file cannot be converted to a .COM file.

Procedure

To convert an executable file to a command file, enter the command

exe2bin *filename filename*.com

where filename is the name of the file to convert.

EXE2BIN is an external command; using it requires having the DOS program EXE2BIN.EXE on a disk in your computer. If you use DOS 2, it must be

either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:exe2bin a:convtdat
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\exe2bin convtdat
```

If the file is larger than 64K, you will see the message `Insufficient memory`. If the file is smaller than 64K, but still cannot be converted, you will see the message `File cannot be converted`.

Warnings

EXE2BIN.EXE does not appear as part of PC-DOS 3.3 or IBM DOS 4.0, although it has been included in every other version of DOS since 1.1. To obtain the version of EXE2BIN.EXE that is compatible with the later IBM releases, you must purchase IBM's *DOS Technical Reference* for that version of DOS.

Exist. *See* If

Exist is a modifier for the If batch file command.

Exit

Syntax

```
exit
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Exit command removes a secondary command processor memory and returns you to the previous command processor.

Procedure

Many programs let you execute DOS commands without ending the program; most do so by loading a secondary command processor (see

Command). You can usually tell if this is the case because the DOS sign-on message (the name and version of DOS) appears on the screen. There are exceptions, however.

If you gain access to DOS by this means, you must type `exit` to return to your application program. Use the same technique if you load a secondary command processor at a DOS prompt.

In DOS 2, control is turned over permanently to the second batch file when you call a batch file from another batch file. This is true even if you use `command /c` to call the second file. The only way to return to the calling file is to end the called file with an Exit command.

Examples

CLOCK.BAT, shown below, loads a TSR that displays a clock in the upper left corner of the screen as well as indicators that show whether the Caps Lock, Num Lock, and Scroll Lock keys are on or off. It also uses ANSI escape sequences to set up a special prompt (see ANSI.SYS). This prompt displays the date in white on red on the top line and extends those colors to the upper right corner, thereby establishing colors for the clock display. Because CLOCK.BAT ends with an Exit command, execution returns to the AUTOEXEC.BAT file after this program and prompt are loaded.

CLOCK.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands in the file
`prompt ←[1;37;41m←[s←[1;50H $d ←[u$p$g ←[33;44m`	Creates the prompt: sets prompt colors to intense white on red; saves current cursor position; moves cursor to top row, 50th column; displays date; returns cursor to saved position; establishes directory prompt; sets screen colors to intense yellow on blue
`\util\clock`	Calls the clock program
`echo .←[1B`	Places a period on the screen to create a line feed after the clock initialization message; moves the cursor two lines down
`cd\`	Makes the root directory current
`exit`	Returns control to the calling program

Tips

In DOS 3.0 or later, you don't need the Exit command in batch files. When you execute a secondary batch file with `command /c` or `call`, control automatically returns to the calling program.

For related information, see AUTOEXEC.BAT File, Batch Files, and Command.

Extensions

Overview

File names may have an optional three-character *extension*. Extensions generally identify the type of file to which they are attached. The extension is separated from the file name by a period.

DOS identifies several types of files by their extension. Other files of the same type use the same extensions. Many software packages identify the files they create by automatically attaching a specific extension to them. However, when using programs that do not require their files to have specific extensions, you can use the extra three characters to add additional information to the file name. For example, you might use the extension .LTR for letters, .MMO for memos, and .RPT for reports.

DOS reserves the following extensions:

.COM	Command files (executable files)
.EXE	Executable files
.BAT	Batch files (see Batch Files)
.SYS	Device drivers (see CONFIG.SYS File)

From the user's point of view, the only practical difference between .COM and .EXE files occurs when a .COM file is in the same directory as an .EXE file of the same name. DOS, under these circumstances, executes only the .COM file. .COM files will not run under OS/2.

Table 15 lists some common file name extensions and their meanings.

TABLE 15

Common File Name Extensions

EXTENSION	MEANING
.ASM	Assembly-language source-code
.BAK	Backup files
.BAS	BASIC program files
.BIN	Binary files
.C	C language source-code files
.COB	COBOL source-code files
.DAT	Data files
.DOC	Document (text) files
.FOR	FORTRAN source-code files
.H	C language header files
.LIB	Compiler library files
.LST	"List" files; may be printable byproducts of a program-compilation process, or word-processor "print-to-disk" files
.MAP	A cross-reference table created during the process of linking program object code
.OBJ	Object code resulting from compiling a program
.OVR, .OVL	Overlay files, containing program code to be swapped into and out of memory by the accompanying program, to minimize the memory required to run the program
.SYS	Device drivers
.PAS	Pascal source-code files
.PIF	Program information file (used by Microsoft Windows and some other programs)
.TMP	Temporary files
.TXT	Text files
.$$$	Temporary files

FastOpen

Syntax

{1} fastopen *drive*[=*nnn*] *drive*[=*nnn*] . . .
{2} fastopen *drive*=(*nnn*,*sss*) *drive*=(*nnn*,*sss*) . . .
{3} fastopen *drive*=(*nnn*,) *drive*=(*nnn*,) . . .
{4} fastopen *drive*=(,*sss*) *drive*=(,*sss*) . . .
{5} fastopen *drive*=(*nnn*,*sss*) drive=(*nnn*,*sss*) . . . /x
{6} fastopen *drive*=(*nnn*,) *drive*=(*nnn*,) . . . /x
{7} fastopen *drive*=(,*sss*) *drive*=(,*sss*) . . . /x
{8} install=fastopen.exe *drive*=(*nnn*,*sss*) *drive*=(*nnn*,*sss*) . . .
{9} install=fastopen.exe *drive*=(*nnn*,) *drive*=(*nnn*,) . . .
{10} install=fastopen.exe *drive*=(,*sss*) *drive*=(,*sss*) . . .
{11} install=fastopen.exe *drive*=(*nnn*,*sss*) *drive*=(*nnn*,*sss*) . . . /x
{12} install=fastopen.exe *drive*=(*nnn*,) *drive*=(*nnn*,) . . . /x
{13} install=fastopen.exe *drive*=(,*sss*) *drive*=(,*sss*) . . . /x

DOS Versions

{1} MS-DOS 3.3 to 3.31
{1} PC-DOS 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

The FastOpen command creates a buffer to hold file names and path names used in commands. It is a TSR program that reduces the time needed for DOS to find files or directories that you have already used (see TSRs).

In DOS 4.0 you can load it in low memory by using the Install= command in your CONFIG.SYS file. The DOS 4.0 version also optionally sets aside a number of *continuous space buffers,* which speed up access to files that are fragmented on disk (see ChkDsk and File Allocation Table for information on file fragmentation).

FastOpen uses about 3K of RAM, plus 40 bytes for each path it stores. You may have to experiment to find the buffer size that encroaches least on your application programs but still increases the speed of file searches.

When the FastOpen buffer becomes full, the least recently used path name or file name is deleted to make room for the next one you use.

Procedure/ Examples

FastOpen is an external command; using it requires having the DOS program FASTOPEN.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\fastopen
```

To install FastOpen in DOS 3.3, enter the command `fastopen` optionally followed by the names of the hard disks for which you want it to remember path and file names. If you do not tell FastOpen how many path and file names to remember, it defaults to 100 for each disk. You can vary this number from 10 to 999 by specifying the number when you invoke the command. To do so, enter a command such as `fastopen c:=60`.

There are several ways to install FastOpen in DOS 4.0. If you enter the command with no parameters, it sets aside space for 50 path and file names and 25 continuous space buffers for each hard disk. If you want to specify that FastOpen buffer path and file names, but not create continuous space buffers, you can enter a command such as

```
fastopen c:=50 d:=20
```

or

```
fastopen c:=(50) d:=(20)
```

or

```
fastopen c:=(50,) d:=(20,)
```

These commands all tell FastOpen to reserve space for 50 path and file names from drive C and 20 from drive D. You can even mix the three forms; they have the same effect.

The DOS 4.0 version of FastOpen also creates buffers of a second type. These buffers supplement the buffers created by the Buffers = command, but have an additional advantage. They store the parts of a file contiguously. This is very helpful if your files are heavily fragmented on disk, because it speeds up access to them. If you want to change the number of continuous space buffers from the default of 25, you do so by entering a second parameter for each drive:

```
fastopen c:=(50,25) d:=(20,10)
```

You can reserve continuous space buffers, but skip the file name buffers, by omitting the first parameter:

```
fastopen c:=(,25) d:=(,10)
```

In addition to these options, you can have FastOpen store its buffers in expanded memory if you have one of three expanded memory boards approved by IBM, and you load the XMA2EMS.SYS device driver with the

appropriate parameters. To do so, add the /x switch to the end of the command. See Memory, XMAEM.SYS, and XMA2EMS.SYS for further details.

You can install FastOpen in low memory, which makes more efficient use of your RAM, by using an Install= command in your CONFIG.SYS file. You must specify the full path to the file if it's not on the root directory of your boot disk, and you must include the extension; for example:

```
install=c:\dos\fastopen.exe c:=60 d:=(20,10)
```

See Install= and CONFIG.SYS File for details.

Warnings

You can invoke FastOpen only once each time you boot your computer. Therefore, you should take care to specify all the drives you want it to buffer when you invoke it. You can include as many hard drives as you like on the command line, but you cannot include diskette drives. The allowable number of buffers also depends on the version of DOS. With PC-DOS you may have from 1 to 99 buffers; with MS-DOS releases through 3.3, you may have from 2 to 255 buffers. With DOS 4.0 you can have up to 10,000 buffers if you also use one of the expanded memory boards that can be addressed by XMA2EMS.SYS (see Memory and XMA2EMS.SYS). To use this larger number of buffers, you must first load the XMA2EMS.SYS driver in your CONFIG.SYS file, configured appropriately, and then add the /x switch to the FastOpen command. (If you use the Install= command to load FastOpen, you must place it after the Device= command that loads XMA2EMS.SYS.) You can place the buffers in expanded memory with this switch no matter how many buffers you establish.

If you use the /x switch with a memory board that is not 100 percent compatible with one of the three memory boards manufactured by IBM for which the XMA2EMS.SYS driver is designed, you may find your root directory and/or file allocation table wiped out the first time you access a disk. I strongly suggest that you do not use this switch if you're not positive your board is designed for it. A list of these boards appears in the XMA2EMS.SYS entry.

Tips

If you use a utility program to unfragment your files (a process generally called optimizing your disk), you probably don't need continuous space buffers, and can save memory by omitting the second parameter from the FastOpen command.

FC

Syntax

fc [*/switches*] [*drive*][*path*\]*file1.ext* [*drive*][*path*\]*file1.ext*

DOS Versions

MS-DOS 2.1 to 4.0

Overview

FC invokes the MS-DOS file-comparison utility, which compares files and reports differences between them. FC allows comparison between two individual files or two sets of files specified by wildcard characters. You can compare files byte by byte or line by line; the former is appropriate for program files, the latter for text files. Aspects of the comparison are controlled by using combinations of these seven switches:

SWITCH	RESULT
/a	Abbreviates the results of a comparison of ASCII files. When FC encounters a group of lines that differ between the two files, it displays the beginning and ending line of the section that is different. The lines between the beginning and ending lines of the differing sections are replaced by ellipses.
/b	Compares binary, or nontext files.
/c	Ignores the case of letters when comparing.
/l	Compares files as ASCII (text) files. This is the default if you don't use a switch.
/lb *nnn*	Compares files as ASCII files, with a memory buffer that can hold *nnn* lines at a time. The default is a 100-line buffer if you use no switches. If the files are too large to fit in this buffer, and no match is found in the part of the files that are in the buffer, the command will terminate with a message. Increase the size of the buffer if this happens.
/n	Displays each differing line preceded by its line number.
/t	Does not convert tab characters (ASCII 9) to spaces. Without this switch, a tab character is replaced by enough spaces to place the following character at a screen tab stop. Screen tab stops occur every eight columns.
/w	Compresses white space (tabs and spaces) to a single space. The presence or absence of white space between words will still be noted, but its size will be ignored.
/nnnn	Specifies the number of lines that must match after a difference is found. If fewer than this number of lines match after a difference, the lines will be displayed as differences.

Procedure/ Examples

FC is an external command; using it requires having the DOS program FC.EXE on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:fc /w a:thisfile.txt a:thatfile.txt
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\fc /b test1.com test2.com
```

If either file is not in the current directory, you may precede it with a drive and/or path name.

You can compare groups of files by using wildcard characters in the file names (see Wildcard Characters).

Comparing Text Files To compare text files, enter the FC command followed by the names of the files to compare; for example,

```
fc report.txt c:\reports\report.txt
```

If the files are the same, FC will report `Files compare OK`. If not, it will report their differences, as shown in Figure 54. As you can see from the command line at the top of the figure, FC is comparing two batch files.

Let's examine the difference reporting closely by looking at the first comparison line by line:

```
***** ltr.bat
CD\%1
COPY %2.LTR D:
D:
***** ms.bat
CD\%1
COPY %2. D:
D:
*****
```

As you can see, FC displays a portion of each file each time FC finds a difference. For each comparison, the following four items appear:

1. The name of the file, preceded by asterisks
2. The line before the one containing text that differs from the other file
3. The line containing the difference
4. The line after the line containing the differing text

```
C:\BATCH>fc ltr.bat ms.bat

***** ltr.bat
CD\%1
COPY %2.LTR D:
D:
***** ms.bat
CD\%1
COPY %2. D:
D:
*****

***** ltr.bat
BREAK OFF
WSLTR %2.LTR
BREAK ON
***** ms.bat
BREAK OFF
WSMS %2.
BREAK ON
*****

***** ltr.bat
NO *.BAK REPLACE D:*.* C: /D
REPLACE D:*.LTR C: /A/P
MIRROR C:>NUL
***** ms.bat
NO *.BAK REPLACE D:*.* C: /D
REPLACE D:*. C: /A/P
MIRROR C:>NUL
*****

C:\BATCH>_
```

Figure 54: Comparing two batch files with FC

The comparison is terminated by another row of asterisks. FC reports the difference the same way each time it finds a difference between the two files.

Comparing Text Files Using Line Numbers You may want your comparisons to show the numbers of the lines where differences occur. Use the /n switch to do so. Figure 55 shows a comparison of the same two batch files, with the lines numbered, which makes it easier to locate differences. In the example, the files are relatively short. However, the figure shows how the FC command formats output when you use the /n switch.

Comparing Binary Files Use the /b switch to compare binary files. Figure 56 shows a comparison of two small .COM files that were forerunners of the FRIDAY.COM program shown in the ErrorLevel entry.

```
C:\BATCH>fc/n ltr.bat ms.bat

***** ltr.bat
    6:  CD\%1
    7:  COPY %2.LTR D:
    8:  D:
***** ms.bat
    6:  CD\%1
    7:  COPY %2. D:
    8:  D:
*****

***** ltr.bat
    9:  BREAK OFF
   10:  WSLTR %2.LTR
   11:  BREAK ON
***** ms.bat
    9:  BREAK OFF
   10:  WSMS %2.
   11:  BREAK ON
*****

***** ltr.bat
   13:  NO *.BAK REPLACE D:*.* C: /D
   14:  REPLACE D:*.LTR C: /A/P
   15:  MIRROR C:>NUL
***** ms.bat
   13:  NO *.BAK REPLACE D:*.* C: /D
   14:  REPLACE D:*. C: /A/P
   15:  MIRROR C:>NUL
*****

C:\BATCH>_
```

Figure 55: Comparing two batch files with FC, using line numbers

```
C:\PROGTEST>fc /b dateis.com monthis.com


00000004: B4 88
00000005: 4C F0
00000006: 88 B4
00000007: D0 4C

C:\PROGTEST>_
```

Figure 56: Comparing program files with FC

The /b switch forces a byte-by-byte comparison. For each difference found, FC reports three items:

1. The offset from the beginning of the file (that is, the number of bytes after the beginning) at which a difference is found, noted in hexadecimal
2. The hexadecimal value of the byte at that offset in the first file
3. The hexadecimal value of the byte at that offset in the second file

In this comparison, only four bytes differed between the two files.

Warnings

If your files are too big to fit in the default buffer, and no similarities are encountered by the time the entire buffer is read, FC will display the message `Resynch failed. Files are too different`. You can try again, using the /lb *nnn* switch with a number higher than 100.

Tips

The white-space switches, /s and /w, are useful for comparing files that you suspect have the same text, but that may be formatted differently.

The FC command with the /b switch is especially useful if you suspect that one of your program files has become corrupted. If one of your programs is misbehaving, compare the copy you are using with the copy on the distribution diskette. If you find differences, copy the file from the distribution diskette to your working disk.

This may not work if an installation program altered the questionable program in some way while installing it. In any case, you should have made a backup right after installing the program. If you did, use that copy for comparison.

The PC-DOS file-comparison command is Comp. See that entry if you use PC-DOS rather than MS-DOS.

For tips on multiple comparisons using wildcard patterns, see Comp.

FCBs=

Syntax

`fcbs=`*`maximum`*`,`*`stayopen`*

DOS Versions

MS-DOS 3.1 to 4.0
PC-DOS 3.1 to 3.3
IBM DOS 4.0

Overview

FCBs= is a configuration command, which you may use in your CONFIG.SYS file. The FCB (*file control block*), is a method by which DOS accesses files. An FCB is a block of memory set aside by DOS that contains information on the name, location, length, and structure of the file to which it pertains. It is a holdover from the CP/M system of structuring files in 128-byte

records. FCBs were replaced in DOS 2 by a new system derived from UNIX that keeps track of files using file handles.

If a program does not recognize path syntax, it probably uses FCBs and the CP/M style of file structure. Late releases of DOS include the FCBs= command for two reasons:

1. To allow older programs that use large numbers of FCBs to run in the new environment
2. To protect files opened with FCBs from being arbitrarily closed when a network is loaded

Procedure

FCBs= is a configuration command entered by means of the CONFIG.SYS file. To activate it, enter a command of the form

```
fcbs=maximum,stayopen
```

in your CONFIG.SYS file. In this command *maximum* represents the maximum number of files that can be opened using FCBs, and *stayopen* represents the number that cannot be closed. When that number is reached, DOS arbitrarily closes the FCB for the least recently used file. The default for each parameter is 4, but you can set either one anywhere from 1 to 255. The second parameter should never be greater than the first.

This command should follow commands that load device drivers.

You have about 40 bytes less available RAM for each FCB above 4.

Warnings

If your computer is not on a network, the only circumstance in which you will need a FCBs= command is if a program that worked with DOS 2 doesn't work with DOS 3 or 4, and if that program doesn't recognize path syntax.

Programs with these two characteristics use FCBs rather than the newer file handles. You will need to increase the number of FCBs available to the program for it work again.

Tips

For related information, see CONFIG.SYS File and Files=.

FDisk

Syntax

```
fdisk
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The FDisk command prepares your hard (*fixed*) disk for DOS. The FDisk program creates a *DOS partition* — that is, a portion of your hard disk to be used for DOS programs. It can optionally create partitions for other operating systems. It also determines which partition (that is, which operating system) will be active when you boot the computer. Most MS-DOS users assign the entire disk to DOS. However, if you want to use CP/M-86 or UNIX, for example you can create a partition for it with FDisk.

You can create up to four partitions on a hard disk. With DOS 3.2 and earlier, each partition must be assigned to a different operating system. With DOS 3.3 and later, the partitions can represent separate logical drives, all of which operate under DOS. The number of logical drives you can have, and their relationship to the main drive, varies with the version of DOS.

You cannot use a hard disk with DOS if it does not have a DOS partition on it. Therefore, your hard disk already has a DOS partition if you are currently using it with DOS. If you have just booted a new computer from drive A and can't get drive C to work, you should probably run FDisk from the diskette in drive A.

For the reasons explained in the File Allocation Table entry, DOS releases through 3.2 can address only up to 32 megabytes on a single disk (see File Allocation Table). Release 3.3 can address larger physical disks, and assign all of them to DOS, but can address logical drives only up to 32 megabytes. Releases since 3.3 assign more space to DOS by letting you create an *extended DOS partition* in addition to the normal DOS partition. You create additional, nonbootable DOS drives within the extended partition.

DOS 4.0 takes a new approach to structuring hard disks. It too creates additional DOS-readable drives by letting you install them in an extended

DOS partition. But by allowing for larger than normal disk clusters and disk sectors (see Disk Clusters and Disk Sectors) it makes it possible to have four logical drives of four gigabytes each, all devoted to DOS.

Before release 3.3, you had several choices if you had a hard disk larger than 32 megabytes. You could:

- Use another operating system for part of the disk.
- Use a commercial utility (or a version of DOS such as those marketed by Compaq and Sperry) that could create an extended DOS partition with separate logical drives.
- Use a commercial utility that patches COMMAND.COM (usually by placing a device driver in your CONFIG.SYS file, but sometimes by patching COMMAND.COM once it's in memory) to address larger drives.

The first course is the safest, but not always practical. When forced to choose between the latter two, using several logical disks is preferable to creating a DOS logical drive larger than 32 megabytes. There are two reasons: First, altering COMMAND.COM may yield results you won't like. Second, even though you may have forced DOS to address a larger disk, some of your software is probably incapable of doing so, because it won't know how to address the aberrant-sized sectors and clusters.

DOS 3.3 was the first release to allow creation of several legal DOS partitions (several logical drives on the same hard disk). DOS 3.31 and DOS 4.0 let you create a DOS partition larger than 32 megabytes legally. However, your software may still be unable to address the larger disks, because they use an abnormally large cluster size.

The information on the size and nature of the partitions is contained in a *partition table* on track 0, side 0 of cylinder 0. Nothing else appears on that track. The logical structure of the disk begins on track 0 of side 1. For more information, see Hard Disks, Structure of.

Procedure

FDisk is an external command; using it requires having the DOS program FDISK.COM (or FDISK.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example, `b:fdisk`.

If you use DOS 3.0 or later, and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example, `c:\dos\fdisk`.

FDisk is entirely menu driven. When you invoke the command, you will probably see the standard menu shown in Figure 57. Details may be somewhat different, depending on your version of DOS and the size of your hard disk.

If you are setting up a hard disk for the first time with DOS 4.0, the Select program will optionally run FDisk for you, but you will still have to go through all the steps it requires.

The choices available with each option depend on the current state of your hard disk. For example, if your hard disk has never been partitioned or does not have a DOS partition, secondary menus will differ from those with a disk already entirely assigned to DOS.

If you have DOS 3.3 or later, you will see an additional option on the main menu:

```
5. Select Next Fixed Disk Drive
```

This option allows you to work with a hard disk other than drive C. This refers only to additional physical drives and not to additional logical drives that may coexist on your first physical drive.

Note: Any files in a partition will be completely erased if you remove it. If you invoke the Create Partition option when your disk already has a DOS partition, you will see a message such as `DOS partition already exists` and you will be sent back to the main menu. In DOS 3.3 and later,

```
FDISK Options

Current Fixed Disk Drive: 1

Choose one of the following:

    1. Create DOS partition
    2. Change Active Partition
    3. Delete DOS partition
    4. Display Partition Information



Enter choice: [1]

Press ESC to return to DOS
```

Figure 57: The FDisk Main Menu

you may see an additional menu, such as the one shown in Figure 59, if your hard disk has space that is not assigned to the DOS partition. The wording differs slightly from DOS 3.3 to 4.0.

Displaying Partition Information To check the status of your hard disk, choose option 4, Display partition information. This option produces a screen similar to that shown in Figure 58. The screen shows partition information for a 30 megabyte hard disk devoted entirely to DOS. Any other partitions are listed by number in the first column. Information regarding their status, type, starting cylinder, ending cylinder, and size in cylinders also appears.

This screen reports that the disk already has a DOS partition, partition number 1. The status of A means that it is the active (bootable) partition. The Type column shows that it is assigned to DOS. (The PRI code occurs in versions of FDisk included with DOS 3.3 and later, indicating that it is the primary DOS partition.) The other figures show that the DOS partition takes up the entire disk.

Creating a DOS Partition If you cannot now use your hard disk with DOS, you must create a DOS partition. You can also create a DOS partition after removing the current one. You may want to take these steps when you upgrade from one version of DOS to another, especially if your new

```
Display Partition Information

Current Fixed Disk Drive: 1

Partition Status   Type  Start  End Size
 C: 1        A    PRI DOS     0  731  732



Total disk space is  732 cylinders.



Press ESC to return to FDISK Options
```

Figure 58: The Partition Information screen

DOS can handle larger disks than your old one, and your hard disk is larger than 32 megabytes.

Assuming you have DOS 3.2 or earlier and a hard disk no larger than 32 megabytes, choosing option 1, Create DOS partition, will have one of two results. If your disk is usable, you will see the message `DOS partition already exists`. Unless you want to make some major changes, press Escape twice to return to DOS.

If your disk has been physically formatted (see Hard Disks, Interleave Factor) but has not yet been prepared for DOS, you will see the following prompt:

```
Create DOS Partition

Current Fixed Disk Drive: 1

Do you wish to use the entire fixed disk
for DOS (Y/N).....................?[Y]
```

If you want to devote the entire disk to DOS, simply press Return.

If you have DOS 3.3 or later, choosing option 1 will present you with the menu shown in Figure 59. Note that with these releases you can have more than one logical drive on your hard disk, even if the disk is smaller than 32 megabytes.

```
Create DOS Partition

Current Fixed Disk Drive: 1


    1. Create Primary DOS partition
    2. Create Extended DOS partition
    3. Create logical drive(s) in the EXTENDED DOS
       partition



Enter choice: [1]


Press ESC to return to FDISK Options
```

Figure 59: The DOS 3.3 Create Partitions menu

You will see this menu only if your disk is larger than 32 megabytes. If so, and you are using your disk, you already have a primary DOS partition. If it does not take up your hard disk, you may want to remove it, using option 3, and create a new DOS partition, covering the entire disk. **This will wipe out all the information on your hard disk**, so back it up completely before proceeding.

If you have a DOS partition that does not take up the whole disk, you can choose option 2, Create extended DOS partition. Option 2 will assign some or all of the remainder of your hard disk to DOS, and you can then use option 3 to create additional logical drives within the DOS partition. If the logical drives are smaller than 32 megabytes, your software will probably remain compatible with the size of the disks.

If you answer no to the question about using the entire disk for DOS, you will see a screen showing something like Figure 60. In DOS 3.3 or later, you will see the same thing if you answer no to the question on using maximum available space for the extended DOS partition.

If you want to use less than the entire disk for the DOS partition (or for the primary DOS partition in DOS 3.3 or later), change the number in the brackets. If you do so, you will be asked for the starting and ending cylinder of the partition. If you have used less than the entire disk for the DOS partition, and you want to use other operating systems on the remainder, you should boot your system from a disk that contains the alternative operating system and use it to set up the remainder of the

```
Create DOS Partition

Current Fixed Disk Drive: 1

Partition Status   Type  Start  End Size

No partitions are installed.



Total disk space is  732 cylinders.
Maximum space available for partition is 732 cylinders.


Enter partition size.............[ 732]


Press ESC to return to FDISK Options
```

Figure 60: The Create Partition menu for an empty 32 Mb hard disk

disk. If you are using DOS 3.3 or later, and you want to use the remainder as an extended DOS partition, choose

```
2. Create extended DOS partition
```

You will then be asked to specify the starting and ending cylinder of the extended partition. On most hard disks, a track contains seventeen 512-byte sectors, or 8704 bytes. Multiply the number of heads your hard disk has by this number to find out how many bytes are in a cylinder. DOS 3.3 does not allow you to divide a partition between two cylinders. As a rule, if you want to use the entire hard disk for DOS, the starting cylinder of the extended DOS partition will be the cylinder after the last cylinder of the primary DOS partition, and the ending cylinder will be the last cylinder on the disk. These values will be supplied as defaults.

Before you can actually use the extended DOS partition, you must create logical drives in it. In DOS 3.3, you can create three logical drives. In DOS 3.31, you can create up to twenty-three, and they can span more than one physical hard disk. Again, you specify their size by entering their starting and ending cylinders. In DOS 3.3, the default values will be the end of the previous logical drive, and the next 32-megabyte boundary, or the end of the disk, if the remainder is less than 32 megabytes. In DOS 3.31, they will be the same as the defaults for the extended DOS partition. You can repeat the procedure until you have filled the entire disk with logical drives.

In DOS 4.0, several new wrinkles have been added. First, as you have already learned, DOS 4.0's 32-bit FAT addresses allow for the creation of logical volumes of up to four gigabytes. Therefore, if your physical disk can hold four gigabytes or less, the defaults supplied for the primary DOS partition will span the entire disk; otherwise the default will be for a partition of four gigabytes. Moreover, DOS 4.0 supplies, and asks for, information about your disk not in numbers of cylinders, but either in percentages of the disk or in megabytes. You can choose either unit for entering the information. The latter choice is complicated by the fact that this version of FDisk includes the message

```
(1 Mbyte = 1048576 bytes)
```

which is not true. The latter number is derived by multiplying 1024 × 1024, while a megabyte is more commonly calculated as 1000 × 1024. Thus, the figures it gives you to work with are erroneous. It will tell you that a hard disk, rated by its manufacturer as 71.3 megabytes, formatted,

is a 68-megabyte disk, and it describes a 128-megabyte disk as a 120-megabyte disk. Keep this in mind when choosing the size of your partitions.

Once you have established your primary and extended DOS partitions, and go on to establishing drives in the extended DOS partition, you can enter your preferred disk size in oversized megabytes or percentages. However, the percentages you use at this point are fractions of the extended DOS partition, not of the entire disk. For example, if you wanted to create four equal-sized logical disks, and entered 25% as the size of the primary partition, you must enter the size of the volumes in the extended partition as 33%.

If, during the course of these activities, you have created a new primary DOS partition, you will be asked,

```
Do you want this partition to be the
active partition (Y/N).....[Y]
```

If you want to boot from this partition of the disk, press Return. If not, press N, and then press Return. If you want to boot from a different partition, see the following section, Changing the Active Partition.

When you finally finish making your selections and return to the FDisk main menu, you will see a message such as

```
System will now restart
Insert DOS disk in drive A:
Press any key when ready . . .
```

Pressing a key will create a DOS partition taking up the portion of the disk you have specified. As you can see, you must continue this command from a diskette, even if you are not already running it from a diskette. DOS will not permit you to exit from FDisk to the operating system if you have made any changes to the partition table, because one of the first things DOS does when it boots the computer is to attempt to read the partition table on the hard disk.

You must then format any DOS partition or logical drive you create using the Format command (see Format). If you want it to be the active partition, use the /s switch to transfer the operating system to the partition. You can also use the /v switch to give the drive a volume label.

Changing the Active Partition If you use more than one operating system, you may need to change the active partition (the partition that your computer boots from). If you use Concurrent DOS XM, or UNIX, for example, you may want your computer to come up in that operating system. You may also want to change from one active partition to another if you have created several logical DOS volumes. To accomplish this, choose option 2, Change active partition, from the FDisk main menu. You will see a menu similar to the one shown in Figure 58, information for each of the partitions, and the prompt

```
Enter the number of the partition you want to make
Active.....................................[_]
```

If you choose a partition other than the DOS partition, that partition will then be the one that boots the computer when you turn it on and controls it thereafter.

Deleting the DOS Partition To change the size of the DOS partition, first delete the current DOS partition. When you upgrade from a version of DOS 2 to a version of DOS 3, it's also a good idea to delete the DOS partition. **Deleting the DOS partition will erase all the data in your DOS partition, so back up your hard disk twice before proceeding!**

You will again see a screen similar to Figure 58, but with the prompt

```
Enter the number of the partition you want to
remove.....................................[_]
```

When you remove this partition, you can either install a new DOS partition of the same or a different size or create one for another operating system.

If you have DOS 3.3 or later, you will also have the choice of deleting the primary DOS partition, the extended DOS partition, or logical drives in the extended DOS partition, just as you did when creating the partitions. You cannot, however, delete the primary DOS partition without first deleting the extended DOS partition, and you cannot delete the extended DOS partition without first deleting all the logical drives it contains.

Warnings

Whenever you create or remove a partition, you lose all the data in the partition. Therefore, you should back up your hard disk before performing any operation with FDisk (other than displaying partition information). Fortunately, you must go through about six menu steps before the FDisk program actually does anything to your disk. You can back out of menus at any time with either the Esc key or Ctrl-Break.

Even though DOS 3.31 and 4.0 allow you to create logical disks that are larger than 32 megabytes, it is generally not a good idea to do so. Larger volumes, as noted, require either sectors larger than the standard 512 bytes, or clusters as large as 8K. Many programs cannot read sectors or clusters of these unusual sizes. The only good reason for making a volume larger than 32 megabytes is to contain a file that is larger than 32 megabytes. Moreover, it's easier to organize your files if you break down your filing area (that is, your hard disk), into smaller, more manageable units.

If you have used DOS 3.31 or 4.0 to partition and format a hard disk with multiple volumes, you will not be able to read the volume in your primary DOS partition if you boot your computer with a version of DOS other than the one with which you partitioned and formatted the disk. This is because the partitioning schemes used by these releases cannot be understood by other DOS releases. Instead, when you try to read that volume, you will see the message `Invalid drive specification`. This problem does not appear to affect volumes in the extended DOS partition.

Tips

For extra safety, delete the FDISK.COM (or FDISK.EXE) program from your hard disk. Run it from a diskette if you need to use it. This will prevent some uninitiated curiosity-seeker from accidentally deleting everything on your hard disk while you're not looking.

File Allocation Table

Overview

The file allocation table is the means by which DOS keeps track of the location of your files on disk. On every DOS-formatted disk, two copies of the file allocation table appear in the sectors immediately following the boot record. DOS maintains two copies because of the file allocation table's importance.

The file allocation table is proportional to the size of the disk because it must be able to reference every cluster on the disk. (See Table 10 in the entry Diskettes, Structure of, for the size of the file allocation table on various types of diskettes.)

The file allocation table stores a reference to each cluster on the disk as a 12-bit, 16-bit, or 32-bit binary number (see Disk Clusters). The 12-bit format was used in DOS 2; with the proliferation of large hard disks, the 16-bit format was introduced with DOS 3. Hard disks created by DOS releases 3.3 and earlier cannot exceed 32 megabytes because the highest number a 16-bit address can hold is 16,384,000. (This figure represents the number of 2K clusters in 32 megabytes.) DOS automatically uses 12-bit addresses for disks up to 16 megabytes, and 16-bit addresses for disks from 16 to 32 megabytes. DOS 4.0, for the first time, introduces 32-bit addresses, in order to address disks that are larger than 32 megabytes. With a 32-bit address, the largest possible number—and therefore the largest number of disk clusters—is 4,294,967,296, which is four gigabytes of 2K clusters.

Each of the addresses in the file allocation table contains one of the following types of information:

- An indication that the cluster to which it refers is not allocated (that is, available for storing data)
- The address of the next cluster in the file
- An indication that the cluster is the last cluster in a file
- An indication that the cluster is "bad" (that is, unusable)

When DOS reads a file, it first gets the starting cluster number from the directory entry. It then looks at that address in the file allocation table to find the next cluster in the file, then at the address indicated there to find the next cluster, and so on, until it finds an end-of-file indicator.

When DOS writes a file, it first looks at the directory to see if a file by that name already exists. If so, it deallocates the file's clusters in the file allocation table. It next looks at the file allocation table to locate the first empty cluster. It writes to that cluster, then checks the file allocation table to find the next empty cluster. It then writes the location of the second cluster in the FAT address of the first cluster. It repeats the procedure for

each successive cluster until the entire file has been written. It then writes the end-of-file indicator in the FAT address of the last cluster.

When DOS deletes a file, it does not do anything to the data clusters where the data is written. It merely deallocates the file's clusters in the file allocation table. This is what makes it possible to undelete files (provided you have the proper utility software on hand).

As you can see, files are not necessarily written to consecutive clusters. You can find out how many *chains* of clusters (groups of consecutive clusters) a file is written to by issuing a command of the form

```
chkdsk filename.ext
```

(See ChkDsk.) As you revise and erase files, your files become progressively more fragmented. As the files become more fragmented, DOS needs more time to read them. With a large hard disk, this can have a serious negative impact on performance.

With a diskette, you can unfragment your files copying them to a blank, formatted diskette using the `copy *.*` or Xcopy command. This will not affect your original diskette, of course, but the files on the new diskette will all be contiguous.

With a hard disk, the only way to unfragment your files using DOS is to do a complete backup, reformat the disk, and restore all the files. Since the files will be written one at a time to a blank disk, they will then be contiguous. However, various disk-optimizing software packages can do the same job in somewhat less time by writing fragmented files to a blank area of the disk, then deallocating the clusters where they were originally located until there are enough contiguous clusters to write the complete files. Because this procedure is somewhat risky, you should do a complete backup before trying it.

Tips

Compaq DOS 3.31 and DOS 4.0 include an option to format hard disks with a larger DOS partition. However, many programs, including hard-disk maintenance programs, cannot yet address a larger partition.

For related information, see Directories and Disk Clusters.

File Attributes

Overview A file has certain attributes that tell DOS how the file should be treated. These attributes may be either set (turned on) or cleared (turned off). A file with no attributes set is a normal file.

The attributes are as follows:

- Read-only
- Hidden
- System
- Volume label
- Subdirectory
- Archive

The attributes are stored in a single byte in the directory entry for each file (byte 11, to be specific), called the *attribute byte* (see Directories). Figure 61 shows the structure of this byte.

As you can see, a single bit in the attribute byte controls each attribute. To set the attribute, set the bit to 1; to clear it, set the bit to 0.

A file may have more than one attribute. If more than one attribute is set, the value of the attribute byte is the sum of the bits that are set. The way in which DOS deals with a file depends on which bits are set.

Normal Files Normal files have an attribute value of 0. They can be read, copied, written to, and erased.

Read-Only Files Read-only files have an attribute value of 1, controlled by bit 0 of the attribute byte. They can be read and copied, but not written to or erased. An attempt to write to them or erase them will result

7	6	5	4	3	2	1	0
NOT USED	NOT USED	ARCHIVE	SUBDI-RECTORY	VOLUME LABEL	SYSTEM	HIDDEN	READ-ONLY
Decimal:		32	16	8	4	2	1
Hex:		20	10	8	4	2	1

Figure 61: The attribute byte

in the message `Access denied`. You can replace or restore them only if you use the /p switch with the commands, and then only if you respond with a Y to the prompt (see Replace and Restore).

You can set or clear the read-only attribute using the Attrib command in DOS 3.0 or later (see Attrib).

Hidden Files Hidden files have an attribute value of 2, controlled by bit 1 of the attribute byte. They do not appear in directory listings and cannot be addressed by any DOS file commands. You cannot remove a directory that contains hidden files. You can check for the presence of hidden files by using the ChkDsk command (see ChkDsk).

The two system files, IBMBIO.COM and IBMDOS.COM (or IO.SYS and MSDOS.SYS) are hidden files. They are also read-only and system files.

DOS 3 allows you to execute hidden files. In early versions of DOS, hidden files cannot be executed.

Many copy-protected software packages allow you to install them on a hard disk by creating hidden files at specific locations on the disk. Do not change the attributes of such files or do anything that changes their location. Otherwise, you will be unable to run your copy-protected software without reinstalling it.

System Files System files have an attribute value of 4, controlled by bit 2 of the attribute byte. Systems files have the same characteristics as read-only files. It is not entirely clear why this additional attribute exists.

The two DOS system files have the read-only, hidden, and system attributes set. Their attribute value is thus 1 + 2 + 4 = 7.

Volume Labels The volume label has an attribute value of 8, controlled by bit 3 of the attribute byte. A volume label is the name you give to a disk with the `format /v` command or, in DOS 3.0 or later, the Label command (see Format and Label). Although it exists as nothing more than an entry in the root directory, DOS considers volume label a hidden file of zero length. If ChkDsk produces a report including the message `0 bytes in 1 hidden file(s)`, the file referred to is a volume label (see ChkDsk). A disk can have no more than one volume label.

Subdirectories Subdirectories have the attribute value of 16 decimal, or 10 hex. Bit 4 of the attribute byte controls this value. Subdirectories take the form of files on disk, although their content is structured like that of the root directory.

Archive Files Bit 5 of the attribute byte is the archive bit. It is set automatically (to 32 decimal, or 20 hex) by DOS whenever a file is created or written to. The Backup and Xcopy commands can be made to use the archive bit in determining which files to copy (see Backup and Xcopy). They can also clear the archive bit so that the command will not back up or copy the same files the next time you issue it.

If a file has never been backed up, its archive bit is set and its attribute value is 10 (hex) higher than it would otherwise be. The Attrib command in DOS 3.2 or later can set or clear the archive bit for any file (see Attrib).

The Copy command always sets the archive bit of the new copy of a file. It does not change the attributes of the original (see Copy).

Procedure

The only DOS command that addresses file attributes directly is the Attrib command, introduced in DOS 3.0 (see Attrib). This command addresses only some of the file attributes. In all versions, you can set or clear the read-only bit with the Attrib command. In releases since 3.2, you can also set or clear the archive bit.

DOS provides no way of addressing the other file attributes directly. However, many commercially available utility programs allow you to alter file attributes. If you have no such utilities, you can still change a file's attributes. To do so, load the directory sector in which its name appears into DEBUG, then write a new value to the 11th (Bth hex) byte of the directory entry. Information on how to do this follows.

Examples

It's not easy to alter an attribute byte using DEBUG, especially if the file you want to change is in a subdirectory. Indeed, it might be simpler to copy the file to the root directory first since the hardest part is locating the sectors containing the subdirectory. To change a file's attributes, follow these steps:

1. First load the root directory into memory using DEBUG's load command. This takes the form

   `l100 drive starting-sector number-of-sectors`

   The *drive* is a number equivalent to the drive letter, assuming that drive A is 0, and *starting-sector* is the number of the first sector of the root directory. The *number-of-sectors* is the number of disk sectors DEBUG loads at one time. Ideally, this should be the entire root directory.
2. Determine the starting sector number to use in this command. This number will depend on the type and size of disk you have. On a hard disk, it is difficult to determine without a utility program; it's also difficult to determine by trial and error. On a diskette, however, you can make an approximation by taking the following steps:
   a. Format a diskette.
   b. Note the number of bytes free in the Format message.
   c. Divide that number by 1024.
   d. Subtract the result from the nominal capacity of the diskette. The resulting number is the number of kilobytes reserved by DOS for the boot record, file allocation tables, and root directory.
   e. Divide the result by 2 to find the number of disk sectors. The root directory will end at the sector with the number that is one less than the result.
   f. Work backward, one sector at a time (more, if you are using a hard disk) until you find the beginning of the root directory. Your task is easier if it's a system disk since the first two directory entries will be the system files.
3. If the file you want to work with is in the root directory, proceed to step 5. Otherwise, locate bytes A and B (hex) in the second line of the directory entry of the subdirectory that contains the file as displayed in DEBUG. (Figure 62 shows part of a directory as listed in DEBUG.) Multiply the hex number in byte A by 100 hex, then add it to the number in byte B (hex). This gives you the starting cluster of the subdirectory that contains the file.
4. Use the Load command again to load the sectors containing the target subdirectory.
5. Find the name of the file you want to work with, then locate byte B in the first line of the directory entry. This is the attribute byte. Enter the new attribute value for the file using the Enter command:

   `e nnnB a`

```
C:\>debug
-d
429E:0480  54 52 45 45 49 4E 46 4F-4E 43 44 00 00 00 00 00   TREEINFONCD.....
429E:0490  00 00 00 00 00 00 B2 4C-2B 10 A5 0E CB 03 00 00   .......L+.......
429E:04A0  56 5F 4D 53 43 49 42 4D-42 49 4E 00 00 00 00 00   V_MSCIBMBIN.....
429E:04B0  00 00 00 00 00 00 47 5D-15 0F 5D 26 50 08 00 00   ......G]..]&P...
429E:04C0  41 43 43 4F 55 4E 54 53-20 20 20 10 00 00 00 00   ACCOUNTS   .....
429E:04D0  00 00 00 00 00 00 00 60-72 0E 35 00 00 00 00 00   .......`r.5.....
429E:04E0  41 44 44 49 53 4F 4E 20-20 20 20 10 00 00 00 00   ADDISON    .....
429E:04F0  00 00 00 00 00 00 11 48-BA 0E C8 06 00 00 00 00   .......H........
-d
429E:0500  42 41 53 49 43 20 20 20-20 20 20 10 00 00 00 00   BASIC      .....
429E:0510  00 00 00 00 00 00 00 60-72 0E 36 00 00 00 00 00   .......`r.6.....
429E:0520  42 41 54 43 48 20 20 20-20 20 20 10 00 00 00 00   BATCH      .....
429E:0530  00 00 00 00 00 00 00 60-72 0E 2C 00 00 00 00 00   .......`r.,.....
429E:0540  42 55 53 49 4E 45 53 53-20 20 20 10 00 00 00 00   BUSINESS   .....
429E:0550  00 00 00 00 00 00 00 60-72 0E 40 00 00 00 00 00   .......`r.@.....
429E:0560  43 41 4C 45 4E 44 41 52-20 20 20 10 00 00 00 00   CALENDAR   .....
429E:0570  00 00 00 00 00 00 00 60-72 0E 46 00 00 00 00 00   .......`r.F.....
-_
```

Figure 62: A portion of a directory displayed in DEBUG

where *nnn* is the first three digits following the colon at the left side of the line containing the file name, and *a* is the new value (in hex) that you want the attribute byte to have.

6. Use the Dump command to ensure that you entered the right value in the right place:

   d*nnn*0

   where *nnn* is the same as it was in step 5. If everything is correct, go on to step 7. Otherwise, go back to step 4, or, if you are working with a root directory entry, step 1.
7. Write the data back to the disk using the Write command. This will be the same as the last Load command you used, except that you will enter a W instead of an L.
8. Exit from DEBUG by entering the Quit command: q.

That's all there is to it. If that's not enough to get you to buy a byte editor or a file-management utility that can address file attributes, you're either a hard-core hacker or a glutton for punishment. Or perhaps you don't want to experiment with file attributes.

Warnings

Never change the attribute value of a volume label or subdirectory. You risk making the disk or the subdirectory unreadable if you do so.

Many copy-protected programs that can be installed on your hard disk may make special use of the attribute byte. Quite a few of these programs

install hidden files at various points on the disk; they may even use illegal attribute values. The programs then look for a particular file with the expected attribute value at a specific location on the disk. Do nothing to change the attribute value of the file or change its location (such as copying it and erasing the original). If you do so, the protected program will not run and you will "lose your installation" (that is, the program will believe it has not been installed properly). You will be unable to reinstall it if the program's publisher allows only one installation.

Tips For related information, see Attrib, DEBUG, and Directories.

File Names

Overview Every file must have a name, and each file in a given directory must have a name different from all other files in that directory.

Procedure The naming of files follows these rules:

1. A file name must be from one to eight characters in length.
2. A file name may have an extension of up to three characters, separated from the file name by a period.
3. A file name may not include any of the DOS command characters (see Command Characters) / \ | . : , * ? < > + = or the space character.
4. A file name may not be the same as any of the reserved device names (see Device Names). However, a file name may begin with a reserved device name, or may be the same as a device name if you add an extension.

See Extensions for details regarding the use of extensions.

Tips There are several approaches to naming files. On the one hand, when you are working with many files it's easier to remember what's in the files if the names are meaningful. On the other hand, programmers prefer one- or two-character names for executable files because they are easier to type. The latter approach is fine until you have several programs each named WP, ED, and LI!

A third approach, especially useful if you have many related data files, is to make each character position in a file name represent some aspect of the information the file contains. For example, suppose you have regional monthly sales and inventory from eastern, western, midwestern, and southern regions. You might give them names such as the following:

E02SLS.RPT
W11SLS.RPT
M09INV.RPT
S05INV.RPT

The meaning of this format is as follows:

- The first character represents the region (E, W, M, or S)
- The second and third characters represent months
- The fourth, fifth, and sixth characters represent the category of report: SLS for sales, and INV for inventory

You can then use wildcard patterns to select all related reports. Thus, the wildcard pattern `e??sls.rpt` would include all eastern region sales reports for every month. Similarly, `?05inv.rpt` would give you all inventory reports for the month of May. And `m12*.rpt` would give both reports for the Midwest for December.

For related information, see Command Characters, Device Names, Extensions, and Wildcard Characters.

Files=

Syntax

`files=`*nnn*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Files= command tells DOS how many files that use file handles can be open at once. File handles are a means by which DOS accesses files. They are derived from a UNIX system of file tracking, and have virtually replaced the older file control blocks, which were carried over into early releases of DOS from CP/M (see FCBs=).

DOS always requires five file handles, one for each of the following:

- The standard input, or keyboard
- The standard output, or screen
- The standard error device (a second, uninterruptible, channel to the screen)
- The standard printer (LPT1)
- The standard auxiliary device (COM1)

DOS needs one additional file handle for each application program that is running, and one for each file opened by an application program.

Procedure

By default, DOS permits eight file handles to be used at one time. You can change this default to any number from five to 255 by adding a command of the form `files=`*nnn* to your CONFIG.SYS file (see CONFIG.SYS File).

Twenty is the maximum number of files that a single application program can open. Many programs, especially database programs, require twenty open files. This requirement is stated in the documentation for those programs.

Tips

Older programs, especially those that do not recognize path syntax, may use an older system of file management using file control blocks (FCBs) instead of file handles. To increase the number of files available to such programs, use the FCBS= command instead of, or as well as, the Files= command (see FCBs=). For other related information, see CONFIG.SYS File.

Filters

Overview

A *filter* is a program that receives data from another program, alters it in some way, and passes it on to another program, the screen, or a file. DOS includes three filter programs:

More — Stops the screen display from scrolling when the screen is full, and waits for a keypress before continuing.
Sort — Sorts columnar data.
Find — Finds specific text in text files.

Examples

You can use several filters in a single pipe. Suppose you had the phone list shown in the Sort entry, and you want to get a list of only the software dealers, sorted by name. To accomplish this, you would use both the Sort and Find filters (in either order). The command

```
Find "Software" phone.dir|sort/+33
```

would produce the result shown in Figure 63.

On the other hand, the command

```
type phone.dir|sort/+33|find "Software"
```

would result in display shown in Figure 64. The difference is that the Find command always starts with a header. This header is among the input to the Sort command in the first command, and is sorted into its proper position. In the second command, the input to the Find command is not a file; instead, it's the output from the Sort command. Consequently, it has no file name, and there is no header.

Tips

For full information on using these filters, see the respective entries and Redirection and Pipes.

```
D:\>type phone.dir|sort/+33|find "Software"
Software                555-5782  Downtown Center Software
Software          1(472)649-3896  Micro Research, Inc.
Software          1(943)545-3611  Software Express

D:\>_
```

Figure 63: Sorting the output of the Find command

```
D:\>find "Software" phone.dir|sort/+33

---------- phone.dir
Software                555-5782  Downtown Center Software
Software          1(472)649-3896  Micro Research, Inc.
Software          1(943)545-3611  Software Express

D:\>_
```

Figure 64: Searching the output of the Sort command

Find

Syntax

{1} find [/n] "*text*" *file1.ext file2.ext . . .*
{2} find /c "*text*" *file1.ext file2.ext . . .*
{3} find /v [/c][/n] "*text*" *file1.ext file2.ext . . .*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

Find is a filter program (see Filters) used to find instances of text in text files. You can search for the following items:

- All lines containing the specified text
- All lines not containing the specified text
- A count of the number of instances of the text in the file
- The same text in several files at once

Find can also indicate the numbers of the lines on which the text is found (or not found).

Procedure/ Examples

Find is an external command; using it requires having the DOS program FIND.EXE on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:find "sample" thisfile.txt
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\find "sample" thisfile.txt
```

To find text in one or more files, enter the Find command, followed by any switches you want to use, the text to find, in quotation marks, and the names of the files to search. For example,

```
find "Enriquez" phone.dir sales.log
```

You can specify as many files as will fit on the command line, but you cannot use wildcard characters (see Wildcard Characters). If the files to be searched are not in the current directory, you can preface them with drive and/or path names:

```
find "Enriquez" a:phone.dir c:\west\sales.log c:\clients\contact.lst
```

The case of the text you are searching for must exactly match that of the text in the files for Find to report a match. Thus, if one of the files contained the text *ENRIQUEZ*, Find would not consider that instance a match and would not report it.

If the text you are searching for includes quotation marks, use two sets of quotation marks to indicate the quotation marks:

```
find ""Hot dog!" he cried" exclamtn.doc
```

Find displays its results on the screen unless you redirect the output to the printer or a file (see Redirection and Pipes).

Finding Text in a File

{**1**} `find [/n] "`*`text`*`" `*`file1.ext file2.ext . . .`

With no switches, the Find command searches all the files you name for the specified text and displays a report. For each file, the report displays a row of hyphens, followed by the file name. Following this header, each line in which the specified text appears is displayed. Figure 65 shows a report of this type.

If you use the /n switch, the report will precede each instance of the search text with its line number in brackets. Figure 66 illustrates a report of this type.

```
C:\BATCH>find "WS" ltr.bat ms.bat

---------- ltr.bat
KEY WS /ML
M_WS
WSLTR %2.LTR

---------- ms.bat
KEY WS /ML
M_WS
WSMS %2.

C:\BATCH>_
```

Figure 65: Output of the Find command

Counting the Instances of Text in a File

{2} `find /c "text" file1.ext file2.ext . . .`

The /c switch gives you a count of the number of lines containing the specified text. It does not display the lines. Figure 67 shows a report from the Find command with the /c switch.

Counting the Lines Not Containing the Text

{3} `find /v [/c][/n] "text" file1.ext file2.ext . . .`

The /v switch tells Find to report only those lines that do not contain the specified text. You can combine it with the /c switch to get a simple count of the lines not containing the text. Or you can combine it with the /n switch to have the lines numbered. You cannot use the /c and /n switches together. Figure 68 shows a search for lines not containing the specified text, with the output in numbered format.

```
C:\BATCH>find /n "WS" ltr.bat ms.bat

---------- ltr.bat
[3]KEY WS /ML
[5]M_WS
[10]WSLTR %2.LTR

---------- ms.bat
[3]KEY WS /ML
[5]M_WS
[10]WSMS %2.

C:\BATCH>_
```

Figure 66: Output of the Find command with line numbers

```
C:\BATCH>find /c "WS" ltr.bat ms.bat

---------- ltr.bat: 3

---------- ms.bat: 3

C:\BATCH>_
```

Figure 67: Counting instances of text using the Find command

```
C:\BATCH>find /v/n "WS" ltr.bat

---------- ltr.bat
[1]@ECHO OFF
[2]IF %DM%#==%true# DM 25 U IB=W/B>NUL
[4]MOUSESYS >NUL
[6]CD\%1
[7]COPY %2.LTR D:
[8]D:
[9]BREAK OFF
[11]BREAK ON
[12]C:
[13]NO *.BAK REPLACE D:*.* C: /D
[14]REPLACE D:*.LTR C: /A/P
[15]MIRROR C:>NUL
[16]CLS
[17]LI /N
[18]MOUSESYS >NUL
[19]M_PLAIN
[20]CD\%1

C:\BATCH>_
```

Figure 68: Finding lines not containing the specified text

Using Find in a Pipe You can direct the output of other commands through the Find command to filter out only those items that interest you, for example, to see a listing of all the subdirectories of a directory without any of the other files. Since every directory entry for a subdirectory name contains the code <DIR>, and since the < character in this code is not legal in file names (see Command Characters), you could produce such a display with the command

```
dir|find "<"
```

To print out this list, change the command to read

```
dir|find "<" >prn
```

Notice that you need not specify the names of files to search when you use Find in a pipe. The text to be searched comes from the previous program in the pipe. For a fuller explanation of this technique, see Redirection and Pipes.

Similarly, you could display only file names and skip subdirectory names by altering the command to read

```
dir|find /v "<"
```

Tips For related information, see Filters and Redirection and Pipes.

For . . . In . . . Do

Syntax

{**1**} for *%%variable* in (*list*) do *command*
{**2**} for *%variable* in (*list*) do *command*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

For is a batch command that creates a loop. It allows you to execute the same command on a group of files or on a group of commands on the same file.

Procedure

{**1**} for *%%variable* in (*list*) do *command*

A For loop repeats an operation in a batch file. The key to the command is *list*. This is a series of items of the same type, separated by spaces, that you specify in parentheses between the command modifiers In and Do. Most commonly, the items are file names, though they can be commands or any other items that DOS recognizes. If they are file names, they may include wildcard characters.

The *variable* is a place holder that represents each item in the list. It is represented by a single letter. The variable is indicated by two percent signs.

The *command* is any valid DOS command. When executing the For loop, DOS executes this command for each item in the list. If the command requires a file name, simply repeat the place holder in the command. Thus, a command of the form

```
copy filename.ext a:
```

would take the form

```
copy %%variable a:
```

in a For loop. Each successive item in the list (presumably a list of file names), would replace *%%variable* each time the loop was executed.

{**2**} for *%variable* in (*list*) do *command*

You can enter a For loop on the command line instead of in a batch file. The only difference is that you indicate the variable with one percent sign instead of two.

Examples

UPDATE.BAT, in the Hard Disks, Backing Up and Restoring, entry, includes two For loops. The program backs up a single hard-disk directory to a diskette. The first For loop deletes any files from the backup diskette that are not also in the directory:

```
for %%f in (a:*.*) do if not exist c:%%f del a:%%f
```

When DOS executes this loop, it looks at every file name in the diskette's directory (represented by `a:*.*`). Then it looks in the hard disk directory to see if there is a file of the same name:

```
if not exist c:%%f
```

If there is no file by that name in the hard disk directory, DOS deletes the file from the diskette:

```
del a:%%f
```

Let's now look at how the variable works. The variable represents each item in the list—in this instance, every file in the diskette's directory. To translate, you might read the command as, "Look at each item (%%) in the group represented by *.*. If this item does not exist on drive C, delete it from drive A."

The second For loop in UPDATE.BAT reverses this procedure. DOS scans the default directory on drive C and copies to the diskette any data file that is not already in the directory:

```
for %%f in (c:*.dat) do if not exist a:%%f copy %%f a:
```

Since the purpose of the program is to ensure that current data is backed up in usable form, the file name *.DAT is used instead of *.*.

Of course, you can substitute whatever file names you need in the list. For example, suppose you have a directory with word-processing, spreadsheet, graphics, and database files related to a single project. You also have some batch files on the directory, however (one of which is a version of UPDATE.BAT), and perhaps some style sheets for your word-processing documents. You don't want your style sheets and batch files on your backup diskette. To keep your backup diskette current, your version of UPDATE.BAT could include a For loop such as this:

```
for %%f in (*.doc *.wk1 .dbf *.pcx) do copy %%f a:
```

The only restriction is that the complete command may not exceed the DOS limit of 132 characters.

To make a version of UPDATE.BAT that will work on any directory and any list of file names, use replaceable parameters (see Batch Files) in the list:

```
for %%f in (%1 %2 %3 %4 %5 %6 %7 %8 %9) do copy %%f a:
```

You can place commands in the list instead of file names, though there aren't too many useful things to do with this technique.

Tips

You cannot execute more than one command in a For loop unless the list is a list of commands. If you want to execute a series of commands on several files, construct a loop using the Shift and GoTo commands instead of a For loop.

For related information, see AUTOEXEC.BAT File, Batch Files, GoTo, and Shift.

Format

Syntax

```
format [drive][/s][/v][/switches]
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

The Format command structures a disk so DOS can read and write to it. It erases any data currently on the disk, writes the boot record and file allocation tables, and reserves space for the root directory (see Boot Record and File Allocation Table). It also scans the disk for bad sectors and locks out any tracks containing them by marking the tracks containing them as unusable in the file allocation table.

With optional switches, you can give a volume label to a disk or transfer the operating system files (COMMAND.COM and the hidden files IBMBIO.COM and IBMDOS.COM, or IO.SYS and MSDOS.SYS) to it (see Label).

FORMAT F

Each release of DOS through 3.3 has coincided with a hardware change allowing a new disk format. Later releases have added switches to the Format command to allow you to use disk formats readable by earlier versions of DOS. You do not need any of these switches, however, to format a diskette to its full capacity in a drive designed for it. Thus, you can format a 360K diskette in a 360K drive with no switches or a 1.44-megabyte diskette in a 1.44-megabyte drive with no switches. Note that you can use the additional switches to format a diskette to less than its full capacity.

Procedure/ Examples

Format is an external command; using it requires having the DOS program FORMAT.COM (or FORMAT.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:format b:
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\format a:/s/v
```

While you are formatting a disk of any type in DOS 2, the message

```
Formatting . . .
```

appears on the screen, to be replaced by a message of the form

```
Format complete

nnnnnn bytes total disk space
nnnnnn bytes available on disk

Format another (Y/N)?
```

The number represented by *nnnnnn* should be the same in both lines and should equal the nominal capacity of the diskette as shown in Table 16, unless the report also displays a line of the form

```
nnnn bytes in bad sectors
```

If you use the /s switch to transfer the system, the message will include a line of the form

```
nnnnn bytes used by system
```

and the number of bytes available will be correspondingly reduced.

DOS 3 substitutes a running count of the tracks and sectors being formatted for the single-word `Formatting . . .` message. It is expressed in heads and cylinders, so that it applies to hard disks as well as diskettes:

```
Head:   0 Cylinder:   36
```

The numbers replace each other on the screen, until the `Format complete` message replaces the counter.

Formatting a Diskette To format a diskette to its full capacity in a matching drive, enter the Format command, followed by the name of the drive in which you want to format the diskette. If the diskette containing the Format program is currently in the drive in which you want to format a different diskette, enter the command with no parameters. The Format command begins by displaying a message such as

```
Insert new diskette for drive A:
and strike ENTER when ready
```

so that you can switch disks in the drive.

Formatting a System Diskette To transfer the operating system to a diskette when you format it, use the /s switch:

```
format a:/s
```

This command copies your DOS's hidden system files and COMMAND.COM to the new diskette after formatting it. This command is equivalent to the following three commands:

```
format a:
sys a:
copy \command.com a:
```

You can start your computer with a disk formatted with the /s switch.

Giving the Disk a Name To give the disk a name, use the /v switch. When the Format command finishes formatting the disk, it displays the prompt `Volume label (11 characters, ENTER for none)?` You can then

enter a name for your diskette. This name will appear any time you use the Dir, Vol, or ChkDsk command. See Label for a discussion of valid disk names.

In DOS 2, the only way to enter a volume label is with the /v switch. The only way to change one is to reformat the diskette with a new label. In DOS 3.0 and later, you can add or change labels with the Label command.

You can use the /s and /v switches in the same command and in combination with any other switches.

Formatting a Diskette to Less than Capacity The Format command is *downward-compatible*. This means that a drive can format a diskette formatted for less than its full capacity as long as the physical medium is the right size and the format is a valid one. Table 16 shows the formats introduced in each version of DOS. For the details of the structure of a disk with each format, see Diskettes, Structure of.

You create formats other than the default format by using various combinations of switches. Be aware that some combinations of switches are incompatible, with no particular rhyme or reason. The one consistency is that you cannot place a volume label on a diskette with eight sectors per track unless you also use the /s switch.

The following list shows the switches that can be used with 5¼" drives to achieve the first four of the formats described in Table 16:

SWITCH	RESULT
/1	Formats one side of a diskette with nine sectors per track (DOS 1.1).
/8	Formats both sides of a diskette with eight sectors per track (320K).
/1/8	Formats one side of a diskette with eight sectors per track (DOS 1.0).
/4	Formats a 360K diskette in a 1.2-megabyte drive. The result may not be readable in a normal 360K drive.
/b	Formats both sides of a 360K diskette with eight sectors per track (320K), with room reserved for the system files (that is, a disk that can be made bootable with DOS 2.0 or later).

DOS 3.3 introduced two new switches for formatting diskettes to less than capacity in all types of diskette drives. Their main purpose, however, is to

TABLE 16 **Diskette Formats Introduced in Each DOS Release**

DOS RELEASE	DISKETTE TYPE	NOMINAL CAPACITY	NUMBER OF TRACKS	SECTORS PER TRACK
1.0	5¼" SSDD	160K	40	8
1.1	5¼" SSDD	180K	40	9
2.0-2.1	5¼" DSDD	320K	40	8
2.0-2.1	5¼" DSDD	360K	40	9
3.0	5¼" DSHD	1.2MB	80	15
3.2	3½" DSDD	720K	80	9
3.3	3½" DSHD	1.44MB	80	18

allow you to format a 720K diskette in a 1.44-megabyte drive. The new switches are as follows:

SWITCH	RESULT
`/t:`*nn*	Allows you to specify the number (*nn*) of tracks.
`/n:`*n*	Allows you to specify the number of sectors per track.

For a 720K diskette in a 1.44-megabyte drive, use `format a:/t:80/n:9`. For a 360K diskette in a 1.2-megabyte drive, use `format a:/4` or `format a:/t:40/n:9`.

DOS 4.0 introduced yet another switch for formatting a diskette to less than capacity, the /f switch. To format any type of diskette to less than the drive's capacity, or to less than the disk's capacity if the disk capacity matches that of the drive, follow the /f switch by the size to which you want to format the disk. For example, `format a: /f:360` will format a 360K diskette in a 1.2-megabyte drive. Similarly, `format a: /f:180` will format a diskette as a one-sided, 180K diskette in a 1.2-megabyte or 360K drive. You can also use the forms `/f:`*nnn*`k` or `/f:`*nnn*`kb`, but the additional information isn't necessary.

You can use the /s and /v switches with any of these commands.

If you use numbers that are not standard for the type of diskette you are formatting, you will see the message `Invalid parameter` and the program will terminate.

Hard Disk Formatting in DOS 2 In DOS 2, the Format command treats hard disks like any other disks. You simply enter the Format command, with the name of the hard disk as its parameter, and it immediately begins formatting the hard disk. It doesn't even wait for you to press a key, because you can't insert a disk in the drive. It's easy to lose a lot of data that way.

Hard Disk Formatting in DOS 3 DOS 3 introduced several safeguards to keep you from formatting a hard disk by accident. When you enter the Format command with a hard disk name as one of its parameters, DOS 3 displays a message such as

```
Warning! All data in non-removable
drive C: will be destroyed.
Proceed with format (Y/N)? [N]
```

If you press Y, and the hard disk has a volume label, you will be prompted with

```
Press ENTER to begin formatting C:
```

If you then press Return, you must respond to the prompt

```
Volume label (11 characters, ENTER for none)?
```

If the disk has a volume label and you do not enter it correctly, the Format command will terminate with the message `Format failure`. Obviously you can still, through persistent wrongheadedness or willful blindness, format a hard disk by accident, but it isn't easy to do so!

Disk Formatting in DOS 4 DOS 4.0 introduced several new aspects to the disk-formatting process. First, it automatically assigns a serial number—an eight-place hexadecimal number, based on the date and time of formatting—to each disk it formats. Second, it includes in its report on the formatted disk's final condition some additional information on the disk, in the form

```
nnnn bytes in each allocation unit
  nn allocation units on disk
```

The term *allocation unit* refers to what is called a *disk cluster* by almost everyone else. If you need to find out the size of a disk's clusters, this is an easy way to do so.

Third, you can now enter a volume label immediately on entering the Format command by including it in the command; for example:

```
format b: /v:2ndarydata
```

However, if you want a space character in the volume label, you must include the entire volume label in quotation marks:

```
format b: /v:"2ndary data"
```

Otherwise, DOS will interpret the text after the space as an additional, unknown, parameter, and the Format command will terminate.

For details regarding hard disk formats, see Hard Disks, Structure of.

Warnings

Always be sure that a disk does not hold any data that you want to save before formatting it. Unless you use one of several commercial utility programs to format your disks, formatting a disk wipes it clean.

Nearly all hard disks contain some bad sectors, even when they are brand new. Do not be alarmed if the Format report shows that some are present.

Bad sectors on diskettes have a tendency to grow. Rather than risk your valuable data, throw away any diskette for which a bad-sector report appears upon formatting.

When you format a disk with DOS 4.0, it writes the volume label to the boot record, as well as to a null file in the directory. If you don't enter a volume label, it replaces this information in the boot record with the text NO NAME. This makes it impossible to make disks formatted by DOS 4 bootable with other DOS versions.

Tips

Many versions of DOS do not actually wipe out a hard disk when you format it; instead, they simply rewrite the file allocation table and erase the root directory. The data is still present, but you won't know where the files are. If your DOS formats hard disks in this manner, you can recover data from a newly formatted disk using a commercial utility program that can read disk sectors directly and write them to a file.

GoTo

Syntax

goto *label*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

GoTo is a batch command that allows you to transfer control to another point in a batch file. Normally, you use it as part of an If statement or as part of a loop (see If).

Except for the DOS 3.3 Call command, the batch programming language does not permit structured programming. In earlier DOS versions, the GoTo statement is the only way to redirect execution to another part of a batch file. The GoTo command tells DOS to go directly to a specific point in the file, marked by a *label*, which is a colon followed by a series of alphanumeric characters.

In DOS 2, a label can have only eight characters in addition to the colon. If you make your labels longer, DOS will be unable to find them and will display the message `Label not found`. In DOS 3, labels may be of any length, but only the first eight characters are significant, additional characters are ignored.

You may want to redirect execution to another part of a batch file in several circumstances:

- To repeat a group of commands
- To divide your batch file into subroutines
- To execute certain commands only under given conditions

Let's look at these three situations separately.

Procedure

Repeating a Group of Commands To repeat a series of commands, put them in a loop. Since there is no way to control how many times the

loop repeats, you must test for an *exit condition*, under which the loop should no longer repeat. To construct a loop of this type requires the Shift and If commands (see Shift and If). The general form of a batch file loop is:

```
:label1
If condition goto label2
command1 %1
.
.
.
commandn %1
shift
goto label1
:label2
```

When a batch file contains a loop of this type, you pass a series of parameters to it on the command line. When the loop reaches the Shift command, the parameter that was just acted on is discarded, and the next command-line parameter replaces %1. The loop repeats until the terminating condition specified in the If statement occurs.

Here is an alternative construction of the loop:

```
:label1
command1 %1
.
.
.
commandn %1
shift
If not condition goto label1
.
.
.
```

In this type of loop, you test for the terminating condition at the end of the loop. As long as the condition has not occurred, execution will repeat, beginning at *label1*. When the condition does occur, execution continues with the next command.

Conditional Execution and Subroutines You can use GoTo statements in a batch file to execute different routines depending on various

conditions. This technique allows you to add a degree of structure to your batch files. The following example assumes several things:

1. Three possible conditions exist (*condition1*, *condition2*, and *condition3*).
2. A different set of commands is to be executed for each condition.
3. *Condition1* is the default condition.
4. Additional code preceding or following the block diagrammed here will be executed in all conditions.

With these assumptions, we would structure the batch file as follows:

```
if condition2 goto label1
rem Skip first block if not default condition (condition1)
rem Execute first block if default condition (condition1)
.
.
.
goto label3
:label1
rem If not default condition, find out which condition applies
rem If condition2 execute this block
rem If condition3 skip this block
if condition3 goto label2
.
.
.
goto label3
rem Code to be executed if condition3 exists
:label2
.
.
.
:label3
rem Code to be executed in all conditions
```

In this instance, if a default condition exists (*condition1*), the first block of code will execute. Execution then bypasses the second and third modules, beginning with *label1*, and proceeds to *label3*.

If the default condition does not exist, the first If statement directs execution to *label1*. At this point, the program checks for a third condition, *condition3*. If this condition exists, the module following the second If statement is skipped. If it does not, this module is executed.

Examples

Controlling the Flow under Varying Conditions ACCOUNTS.BAT, which follows, can be used to set up an accounting program. It illustrates how to direct the flow of control according to varying conditions. The program (ACCOUNT.EXE) is configured to find its data and index files on a RAM disk, drive D. However, it can open files on another drive if the drive is entered as a command-line parameter.

The flow chart in Figure 69 shows the flow of control in ACCOUNTS.BAT under different conditions. The first If statement tests to see whether the hard drive was specified as a parameter. If so, the program executes with the files on the hard disk. If not, execution proceeds to the `:ramdisk` module. This module copies the data files to the RAM disk, executes the program, then copies the modified files back to the hard disk.

ACCOUNTS.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`cd\accounts`	Selects the \ACCOUNTS directory
`if not %1@==C@ if not %1@==c@ goto ramdisk`	Checks for a parameter; if hard disk not specified, goes to `:ramdisk` routine
`account c:`	Executes the program
`goto backup`	Calls the backup routine
`:ramdisk`	
`echo Copying data files to RAM disk . . .`	
`xcopy *.dat d:>nul`	Copies data files to RAM disk
`xcopy *.idx d:>nul`	
`account d:`	Executes the program
`echo Copying files from RAM disk to hard disk . . .`	
`replace d:*.* c: /d>nul`	Replaces modified data files
`rem "xcopy d:*.dat c: /m/v>nul"`	
`rem "xcopy d:*.idx c: /m/v>nul"`	

continues

COMMANDS	COMMENTS
`del d:*.dat`	Deletes data files from RAM disk
`del d:*.idx`	
`:backup`	
`echo ^Gplace backup disk in drive A.`	
`pause`	Prompts user for backup diskette
`if not exist a:\*.dat goto update`	Checks for data files on diskette; if files exist, gives the user a chance to quit
`echo ^GWARNING: data files exist on A.`	
`echo If you do not want to write over these files,` `echo press Ctrl-Break. Otherwise,`	
`pause`	
`:update`	
`xcopy *.dat a: /v`	Copies data files to diskette
`xcopy *.idx a: /v`	
`cls`	
`if exist a:\batch.dat goto copyok`	Checks success of copy; if unsuccessful, displays message and terminates
`echo ERROR: data files were not saved to A.`	
`goto end`	
`:copyok`	If successful, displays message and shows diskette directory
`echo Data files have been saved to A.`	
`dir a:`	
`:end`	
`cd..`	

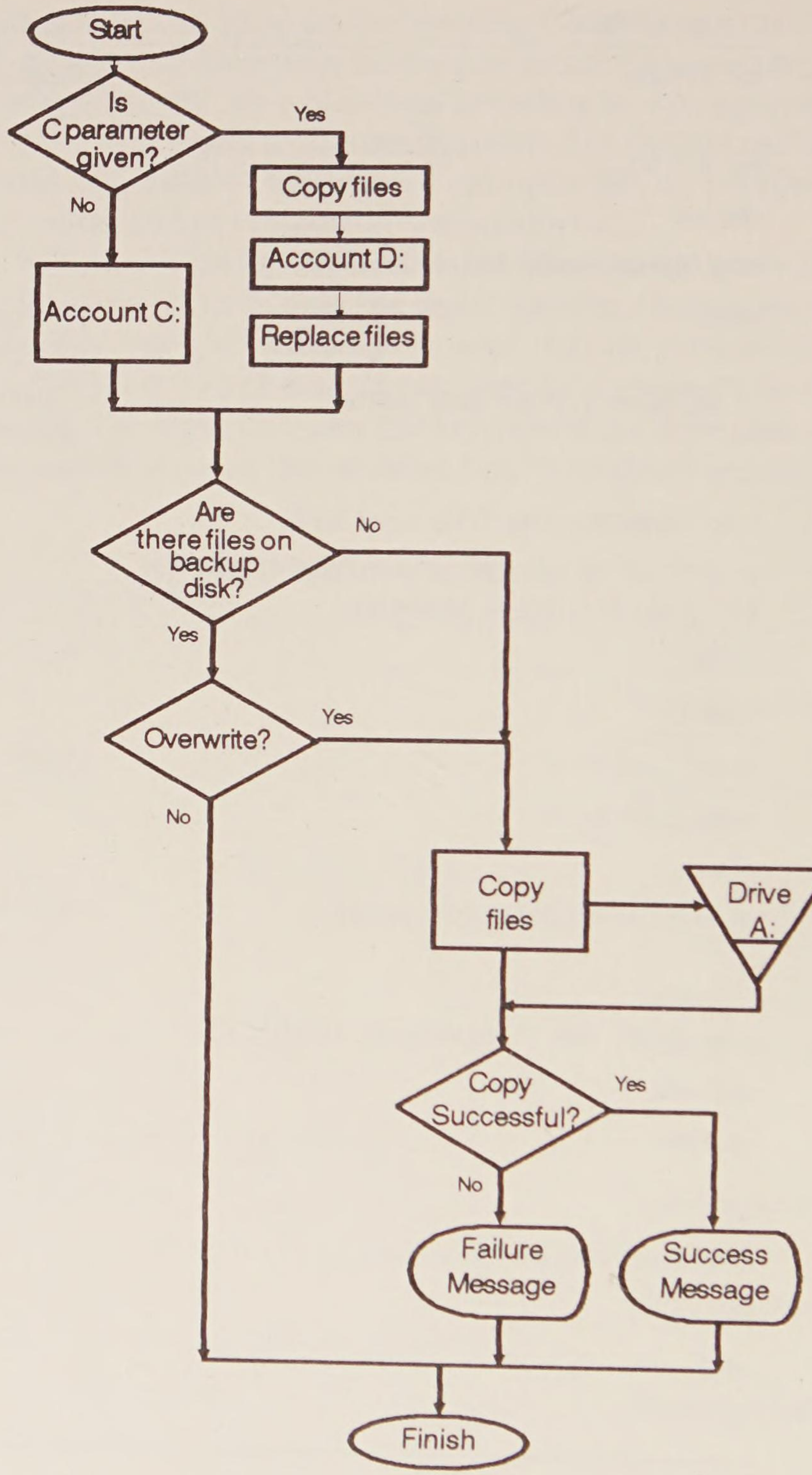


Figure 69: Flowchart for ACCOUNTS.BAT

The Replace command selects which files to copy (see Replace). It copies only files that already exist on the target directory. The /d switch excludes all files except those with a later date or time. (Note that the /d switch for the Replace command is available only in MS-DOS. In DOS 4.0 it has been replaced by the /u switch.) The following two commands, in remarks, will substitute for the Replace command if you use PC-DOS 3.2 or later. The /m switch to the Xcopy command excludes all files except those that have been changed (see Xcopy). However, two commands are necessary since you cannot use a *.* wildcard pattern if there are other files on the RAM disk. If you use DOS 3.1 or earlier, you must use Copy statements, since these releases don't include the Xcopy and Replace commands.

When the program ends, the batch file always backs up the data files to a diskette. The :backup routine itself contains several modules selected by GoTo statements. The routine begins by checking whether there are data files on the diskette in drive B. If there are, execution goes to the :verify routine, where the user decides whether to overwrite the files. A Pause command allows the user to end the program directly, which prevents the files from being overwritten. If the user decides to continue, the :update routine is called.

If there are no data files in drive B, the :update routine executes automatically. When the commands that should copy the files have executed, the program checks to see if the copy was successful. Again, one of two branches is selected, depending on whether the copy was successful.

For another example of a loop using GoTo and labels, see MCOPY.BAT in the Shift entry.

Tips

It's difficult to create true subroutines in batch files because there is no simple way to keep track of the point from which you called a module. As a result, it's hard to return to that point. To avoid this problem, you could set a flag in the environment, then read the flag (see Set). You could then set up a series of If statements to direct execution back to various points depending on the value of the flag. However, this procedure results in extremely slow and clumsy code.

The Call command allows you to call other batch files and to return to the line following the Call statement (see Call). This enables you to use secondary batch files as subroutines. Unfortunately, Call is available only in DOS 3.0 and later.

If you can't use these releases and want to set up programs of this type, you might try a different programming language. Besides the standard programming languages, there are two languages on the market specifically for extending DOS's batch-file capabilities: Personal REXX and Extended Batch Language (EBL). Look into these alternatives if you find yourself frustrated by the batch language.

For related information see Batch Files, If, and Shift.

GrafTabl

Syntax

{**1**} graftabl
{**2**} graftabl *charset*
{**3**} graftabl /sta[tus]
{**4**} graftabl ?

DOS Versions

{**1**}	MS-DOS 3.0 to 3.2
{**1**}	PC-DOS 3.0 to 3.2
{**1**}{**2**}{**3**}	MS-DOS 3.3 to 3.31
{**1**}{**2**}{**3**}	PC-DOS 3.3
{**all**}	IBM DOS 4.0

Overview

The GrafTabl command loads a *graphics table*—essentially a character set—into memory for use with a color graphics or monographics (Hercules-type) display adapter. It serves no purpose with a standard monochrome adapter or an EGA or VGA adapter.

GrafTabl lets you display the high-order ASCII characters (those with codes above 128). In DOS 3.3 and later, it also permits display of one of the alternative language character sets (called *code pages* in the language of IBM and Microsoft). For details on what else you need to do to show alternative character sets, see DISPLAY.SYS, PRINTER.SYS, and Mode.

If your computer was manufactured by a company that sells only in one country, your version of DOS may not include this option.

Procedure

GrafTabl is an external command; using it requires having the DOS program GRAFTABL.COM (or GRAFTABL.EXE) on a disk in your computer.

If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\graftabl
```

{1} `graftabl`

To make it possible to display high-order ASCII characters on a graphics display in DOS 3.2 or earlier, enter the command `graftabl` at a DOS prompt.

GrafTabl is a resident program, and permanently occupies some RAM until you reboot the computer (see TSRs). GrafTabl is not smart enough to know whether it is already in memory. If you invoke it again, the new copy will take up some additional RAM.

{2} `graftabl` *charset*

If you use DOS 3.3 or later, use GrafTabl to prepare your graphics display to display high-order ASCII characters in any of the four national character sets included in DISPLAY.SYS. In this command, *charset* is a three-digit *code page number* from Table 22 in the Mode entry. Do not use 850, the code page number for the international character set. If you do, you will see the message `Incorrect parameter`. If you do not include a character set parameter, GrafTabl loads code page 437, the default (United States) character table.

If all goes well, you will see a message telling which character set (or code page) is loaded. The wording of the message depends on your version of DOS.

GrafTabl is still a TSR in DOS 3.3 and later, but it loads itself only once.

{3} `graftabl /sta[tus]`
{4} `graftabl /?`

In DOS 3.3 and later, the command `graftabl /status` or `graftabl /sta` displays a message telling which character set (code page) is loaded. In DOS 4.0, the command `graftabl /?` displays a similar message, and also presents a list of the available options.

Warnings

Load GrafTabl only once if you use DOS 3.2 or earlier. Doing otherwise wastes memory and serves no purpose.

Tips

GrafTabl has exit codes that you can test with the ErrorLevel command (see ErrorLevel).

For information on using national character sets, see ChCP, DISPLAY.SYS, NLSFunc, PRINTER.SYS, and Mode.

Graphics

Syntax

{**1**} graphics [/r]
{**2**} graphics */printer* [/b][/r]
{**3**} graphics [*printer*] [*profile.pro*] [/b][/r][/lcd][/printbox:*size*]

DOS Versions

{**1**}{**2**} MS-DOS 2.1 to 3.31
{**1**}{**2**} PC-DOS 2.0 to 3.3
{**all**} MS-DOS 4.0
{**all**} IBM DOS 4.0

Overview

The Graphics command loads a TSR that intercepts output to the printer, thereby allowing you to print screen displays in graphics mode on some types of printers (see TSRs). It has no effect on text screens.

To understand the purpose of this command, you must first know something about the way a computer displays images on a screen. MS-DOS computers have a variety of *video modes* (see Mode). Some of these modes are *text modes* and others are *graphics modes*.

In the text modes, characters are displayed in *cells*, which are slightly larger than the characters themselves. These cells are usually arranged in 25 rows of 80 columns. (You can change the default number of columns to 40 on any type of color display. You can change both row and column defaults on EGA, VGA, and monographics [Hercules-type] adapters.)

In the graphics modes, the screen image is *bit-mapped*; that is, the computer codes information separately for each point on the screen that it can recognize (called a *pixel*). In graphics video memory, there is no such thing as a character. You see characters because pixels arranged in a particular shape are in a different color from others. You see characters on a monochrome graphics screen because the appropriate pixels are turned on or off.

GRAPHICS

All printers can print text, and most printers can also print *screen dumps*—row-by-row images of the entire screen—in text mode. Most dot-matrix, thermal, and laser printers can be programmed to print graphics. However, they generally can't print a graphics screen dump without help because the data arriving at the printer from the computer is, as always, nothing more than a stream of bytes. Unless it is forced to do otherwise, the printer normally interprets this stream of bytes as ASCII character codes. Programs that successfully print graphics images do so by loading a *printer driver* file. This file does the necessary conversions and programs the printer to respond correctly to the data. It does not necessarily allow you to print graphics screen dumps, however.

The Graphics command works only with IBM printers, Epson printers with the GrafTrax option, and printers that have an "Epson-compatible" mode. In DOS releases through 3.31, the Graphics command allows you to reproduce a graphics screen display from a color graphics or monographics display. The DOS 4.0 version of the command can also reproduce EGA and VGA screens. In addition, with the use of special switches, it can produce graphics screen dumps on several other types of printers.

In contrast to earlier versions, the DOS 4.0 version of Graphics looks up the information on how to print graphics for each of the types of printers it supports in a *profile file*. By default, this file is called GRAPHICS.PRO, and should be on the same directory as GRAPHICS.COM. This is an ASCII text file that includes remarks. If you have a 9-pin printer that is not among those officially supported by IBM, you can add profiles to GRAPHICS.PRO for your printer, or create an alternative profile file.

Procedure

Graphics is an external command; using it requires having the DOS program GRAPHICS.COM (or GRAPHICS.EXE) on a disk in your computer. If you use DOS 2, it must be in either the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:graphics
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\graphics /color8 /b
```

{1} graphics [/r]

To load the graphics driver, enter the command `graphics` with or without parameters. You need do so only once each time you boot the computer, although doing so again does no harm. When you load Graphics, you have from 4K to 7K less available RAM, depending on your version of DOS.

If your printer can respond to the Graphics driver, you can print graphics-mode screens by pressing Shift-PrtSc. Generally, the screens will appear in black on a white background, sideways on the page. Figure 70 shows a graphics display printed on an Epson printer with Graphics loaded. If you dump the same screen to the same printer without Graphics loaded, the page will move up several lines and nothing will be printed.

Printing Graphics Screens in Inverse Mode If you prefer a white design on a black background—as is normally shown on the computer screen—enter the command `graphics /r`. The /r switch stands for reverse

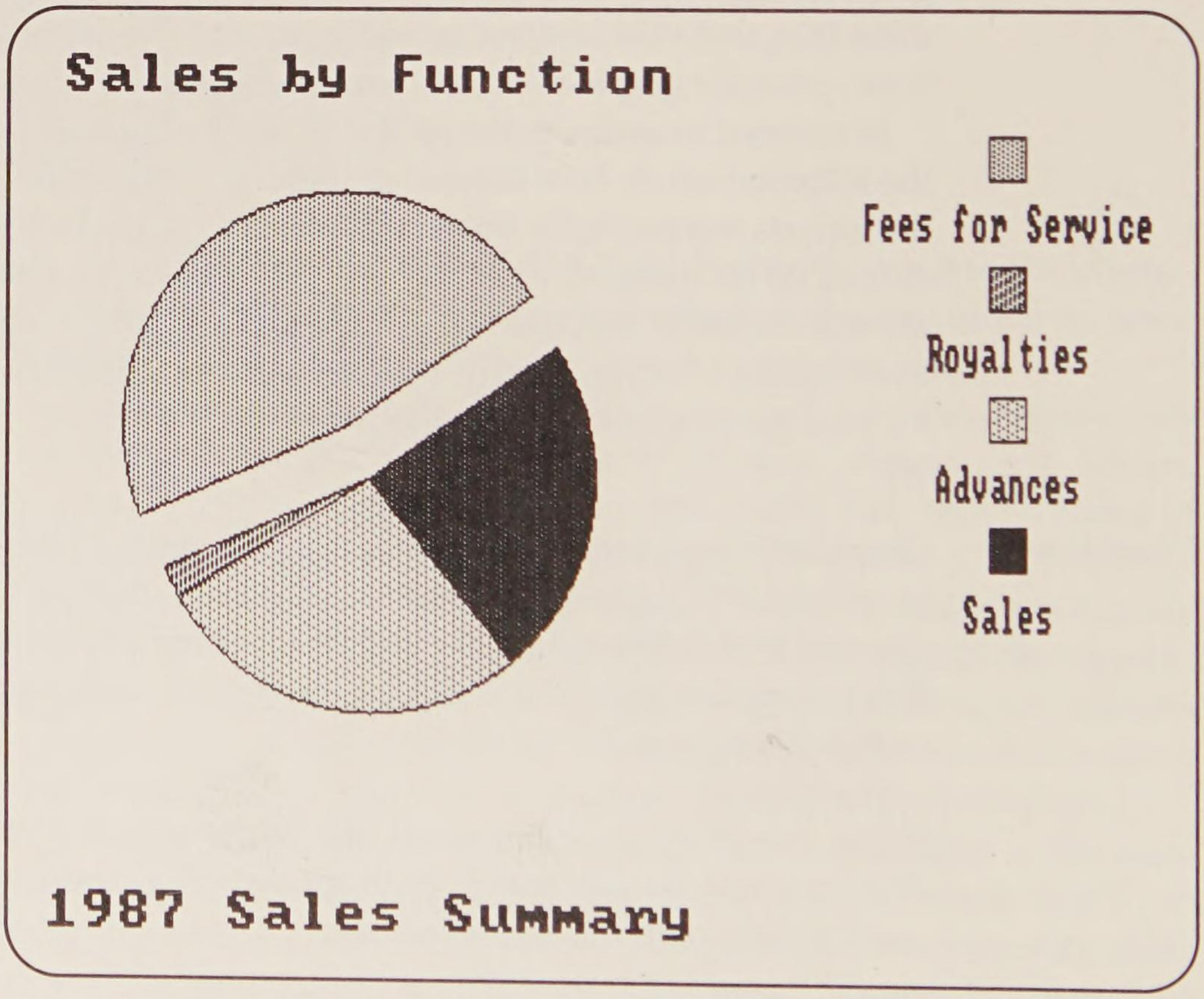


Figure 70: A screen printed with Graphics loaded

video. Figure 71 shows the same screen as Figure 70, printed with the /r option. Once you load Graphics, you can switch to reverse video by entering the Graphics /r command, but you cannot switch back.

Printing Graphics Screens in Color

{2} `graphics /printer [/b][/r]`

If you have an IBM color printer, you can use Graphics to print graphics screens in color. Color/graphics adapters use a maximum of four colors in their displays. You have a choice of *palettes*: red, green, and blue; or cyan, magenta, and yellow, plus a background color. IBM color printers offer a choice of three ribbons:

black
red-green-blue-black
cyan-magenta-yellow-black

corresponding to the graphics palettes of the color/graphics adapter.

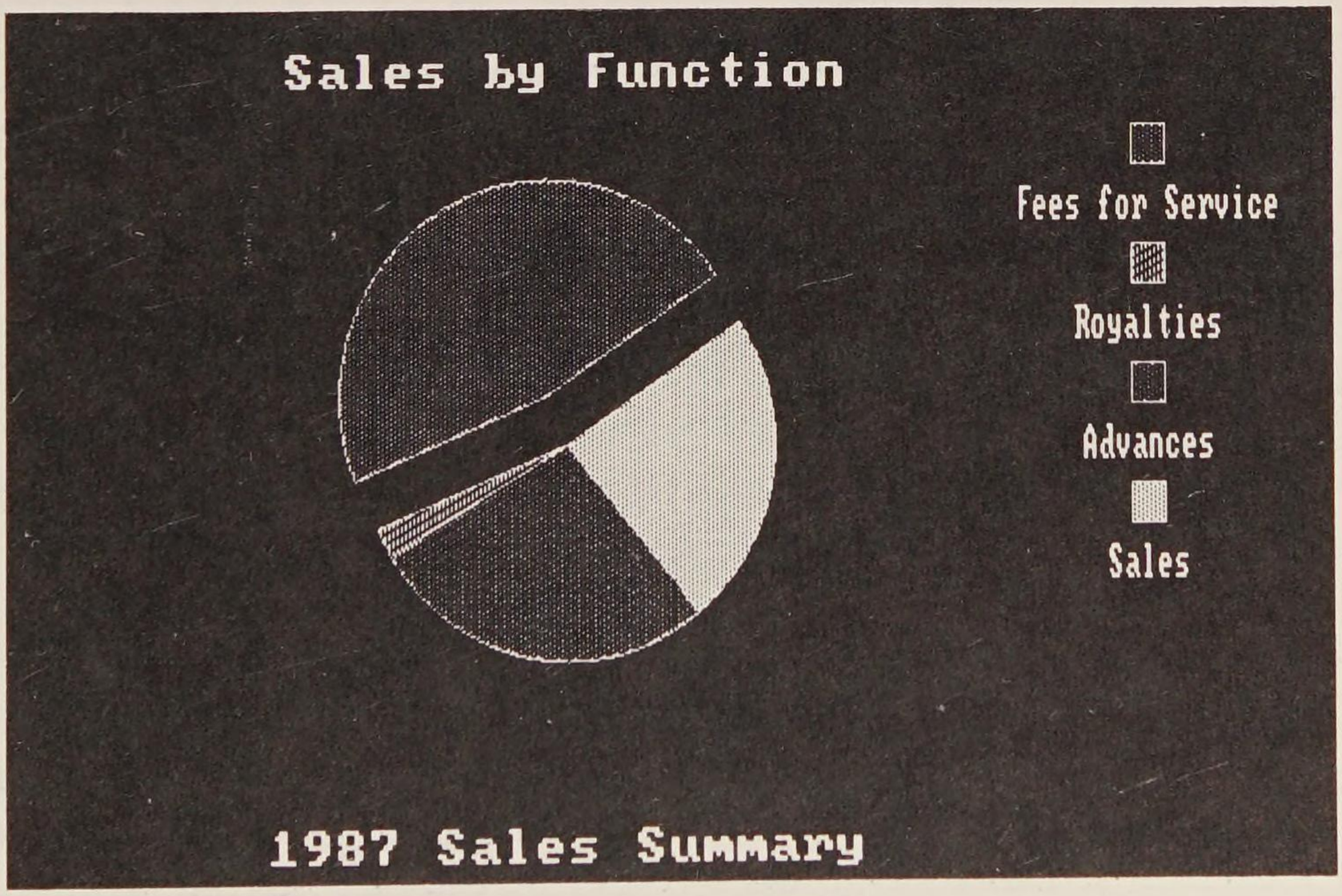


Figure 71: A graphics screen printed with reversed foreground and background

To print graphics screen dumps on an IBM color printer or a compatible printer, you must tell Graphics you are using this type of printer, specifying the color of the ribbon you are using. To do so, enter the appropriate command:

`graphics /color1`	for a black ribbon
`graphics /color4`	for a red-green-blue-black ribbon
`graphics /color8`	for a cyan-magenta-yellow-black ribbon

Because graphics screens on EGA or VGA adapters can have sixteen or more colors, the DOS 4.0 version of the Graphics command uses a table of values in the file GRAPHICS.PRO to map the screen colors to various combinations of the four ribbon colors, to produce eight colors when printing a sixteen-color display with a four-color ribbon. It also uses this file to map the screen colors to up to nineteen shades of gray with a black ribbon. You can adjust the color mapping by editing the GRAPHICS.PRO file and experimenting until you have the results you desire. To ensure that you have a copy of the original settings, copy the file with another name, for example PROFILE.PRO, and then edit the copy. Include the file name in your invocation of the Graphics command; for example,

```
graphics color4 profile.pro
```

If you enter no printer parameter, the printer designation defaults to `graphics /graphics`, which represents the IBM Graphics Printer, the IBM Proprinter, and Epson printers.

When you use color, the /r switch switches black and white, without affecting the other colors. The /b switch controls the background color, which normally is white (unless you use the /r switch).

Printing Graphics Screens on a Wide Carriage The DOS 4.0 version of the Graphics command is capable of taking advantage of the extra wide carriage on the IBM Quietwriter model 5201, the IBM Proprinter models 4202 and 4208, and presumably the Epson printers with a 13½-inch carriage, by printing graphics screens horizontally rather than rotated 90 degrees. The GRAPHICS.PRO file contains information to control the aspect ratio, print density, and whether or not to rotate the image in various graphics modes. You can edit this file to fine-tune the results.

Printing Graphics Screens from the IBM Convertible The DOS 4.0 version of the Graphics command has several special options for printing screens from the LCD display on an IBM Convertible computer. To print the image exactly as it appears on the Convertible's screen, use one of the following commands:

```
graphics /lcd
graphics /printbox:lcd
graphics /pb:lcd
```

(PB is an abbreviation for printbox.) If you want the image to resemble the one produced on a standard-size screen, use the command `graphics /printbox:std` or `graphics /pb:std`. You should also use one of these commands if your Convertible is attached to a standard size screen.

In addition, if you are using the Convertible's thermal printer, you should specify it in your Graphics command; for example,

```
graphics thermal /pb:std /r
```

Editing the Profile File If your printer is not among those supported by DOS 4.0, and it is a 9-pin dot-matrix, thermal, or ink-jet printer, you may be able to write a profile for it. The form of a printer profile is shown in the following example. You enter a line of remarks by preceding it with a semicolon.

```
PRINTER printername
DISPLAYMODE 4,5,13,19     ;; 320x200 > 6.7"x8.9" rotated
  SETup Escape sequence to establish proper line spacing
  GRAPHICS 32,32,32,32,32,32,start graphics mode,LOWCOUNT,HIGHCOUNT
  PRINTBOX STD,4,2,ROTATE
  PRINTBOX LCD,2,2,ROTATE
  RESTORE Escape sequence to restore normal printing

DISPLAYMODE 6,14          ;; 640x200 > 6.7"x8.9" rotated
  SETup Escape sequence to establish proper line spacing
  GRAPHICS 32,32,32,32,32,32,start graphics mode,LOWCOUNT,HIGHCOUNT
  PRINTBOX STD,4,2,ROTATE
  PRINTBOX LCD,2,2,ROTATE
  RESTORE Escape sequence to restore normal printing
```

```
DISPLAYMODE 15,16        ;; 640x350 > 5.8"x8.9" rotated
  SETup Escape sequence to establish proper line spacing
  GRAPHICS 32,32,32,32,32,32,start graphics mode,LOWCOUNT,HIGHCOUNT
  PRINTBOX STD,4,2,ROTATE
  PRINTBOX LCD              ;; PC/Convertible doesn't support these modes
  RESTORE Escape sequence to restore normal printing

DISPLAYMODE 17,18         ;; 640x480 > 8"x8.9" rotated
  SETup Escape sequence to establish proper line spacing
  GRAPHICS 32,32,32,32,32,32,start graphics mode,LOWCOUNT,HIGHCOUNT
  PRINTBOX STD,4,2,ROTATE
  PRINTBOX LCD              ;; PC/Convertible doesn't support these modes
  RESTORE Escape sequence to restore normal printing
```

This calls for some explanation. *Printername* is the name you use as a parameter when you invoke Graphics. It should be a single word, and it doesn't need to be the brand name of your printer. The DISPLAYMODE command refers to the screen mode. Each screen mode has a number. As you can see, the display modes are grouped, so that there is one display mode command for each screen resolution. The PRINTBOX command tells how many horizontal and vertical printer dots should be mapped to each screen pixel. The higher the screen resolution, the higher the number of dots needed to match it. If you don't expect to use an IBM PC-Convertible, you don't need the PRINTBOX LCD command, but you must have a PRINTBOX STD command.

The SETup command is optional. It is often used to enter the escape sequence to set the printer for graphics line spacing. The GRAPHICS command is most important. In it, you enter the escape sequence to set the printer in graphics mode before it accepts a series of bytes. This command tells the printer to interpret the bytes as graphics information codes rather than as ASCII codes. You enter the values in the SETup, GRAPHICS, and optional RESTORE statements as the decimal values of the ASCII codes to be sent to the printer. Thus, if ^[L is the command to set your printer in graphics mode, you would enter 27,76, the ASCII codes for the Escape character and uppercase *L* (see ASCII Code). The 32s in the GRAPHICS statements are space characters, expressed as their ASCII value. They move the print head several spaces to the right before beginning the graphics printing.

Finally, the RESTORE statement issues the codes to return your printer to standard text mode.

For color printers, the additional commands COLORSELECT, COLORPRINT, and DARKADJUST are available to select the color band on the ribbon, to map the sixteen screen colors to the four ribbon colors, and adjust the color saturation of the printout. You specify colors in the COLORPRINT statements by selecting an intensity level for each of the color bands of the ribbon.

If you want to experiment with graphics profiles, you can make a copy of GRAPHICS.PRO, and load it into your word processor in ASCII mode. The worst that can happen is that you'll waste time and paper.

Tips For related information, see Graphics Characters, Displaying and Printing.

Graphics Characters, Displaying and Printing

Overview In addition to the characters on the keyboard, the MS-DOS character set includes many graphic symbols and foreign-language characters (see ASCII Code and Appendix A). Some programs can display some or all of these characters on the screen; others cannot. To use these characters, you must know which characters your program can display and how to enter those characters.

The graphic symbols, also known as *graphics characters*, are of two types: Those with codes from 128 to 254 are called *high-order* characters, and those with codes from 1 to 31, and 127 are called *control* characters. Although graphics characters are assigned to the latter codes, they function primarily to control the computer and its peripherals.

You can enter the high-order graphics characters at the DOS level (at the prompt and in EDLIN). You cannot enter the graphics equivalents of the control characters at this level. However, the Type command can display some of them (see Type).

Procedure/ Examples

Entering Graphics Characters At the DOS level, you can enter any character (including high-order and control characters) in the ASCII character set except 0 and 255. To do so, hold down the Alt key and enter the digits of the character's ASCII code on the numeric keypad. When you lift your finger from the Alt key, the character will appear on the screen. This is a cumbersome way of entering characters for which there are keys, but it's the only way to enter some characters.

You can enter most control characters by holding down the Ctrl key and pressing the key representing the character. The exceptions are the DOS command characters Ctrl-H, Ctrl-I, Ctrl-J, Ctrl-M, Ctrl-P, Ctrl-S, and Escape (Ctrl-[) (see Command Characters). Ctrl-C appears on the screen when you enter it; however, you cannot get it into a file this way, since Ctrl-C cancels the command you are entering.

When you enter control characters by this method at the prompt, they appear as the uppercase character preceded by a caret; for example,

```
^A^B^Z
```

Some application programs let you enter all, or almost all, the ASCII characters using the Alt key and the numeric keypad. Some application programs display the graphics equivalent of the control characters on the screen.

EDLIN has a special technique for entering control characters that cannot be entered with the Ctrl key: Enter a Ctrl-V, then type the uppercase character representing the control character.

Displaying Graphics Characters You may want to use graphics characters to create display screens or menus. The Type command will display all of the high-order graphics characters with no difficulty. For an example, see the ANSI.SYS entry.

The Type command is a bit more persnickety when it comes to control characters. When you use Type to display a file that contains control characters, it will either display the character's graphics representation or respond to the control character. Table 24 in the Type entry lists the control characters that cannot be displayed and tells what effect they have, if any.

When you place these characters in Echo statements in a batch file and execute the batch file, Ctrl-P and Ctrl-[display their graphics equivalents. Ctrl-Z functions as an end-of-file mark in a batch file and, therefore, does not appear.

To test some of the control characters, enter the following lines:

```
copy con test
^A^B^Z
```

(That's Ctrl-A, Ctrl-B, Ctrl-Z.) You should then see the message

```
1 file(s) copied
```

Now enter the command

```
type test
```

You should see two smiling faces, one black and one white. (Again, you won't see the Ctrl-Z because it's the end-of-file mark.)

Some programs, such as Microsoft Word or Framework in graphics mode, let you enter text on a graphics screen. If you use a color/graphics or monographics display adapter, you must enter the GrafTabl command before you can see the high-order ASCII characters (see GrafTabl).

Printing Graphics Characters Printing the graphics characters on most printers is not easy. When you do a screen dump with Shift-PrtSc, most printers do not print graphics characters. Instead, they substitute the equivalent low-order ASCII character (the one with a code 128 less than the high-order character) for any high-order characters. When you print a file containing the high-order characters, they usually appear as their low-order equivalents, perhaps in a different font.

Most printers won't print the control characters in any form. Printers generally respond to some or all of the control characters as commands, and will change some aspect of the print style when they receive one.

Despite these limitations, there are two alternative methods for printing graphics characters on a printer, both of which require external software. One type of software redefines the character codes in your printer to match some or all of the complete MS-DOS character set. The other type of software lets you "capture" a text screen and display it as a graphics screen. You can then print it as a screen dump with the Graphics command loaded (see Graphics).

Figures 72 and 73 show a graph in Framework III, constructed of graphics characters on a text screen. The Framework main screen, or "desktop," is itself constructed of graphics characters. Figure 72 was printed on an Epson printer using the shareware program PR256. This program allows an Epson printer to print all of the high-order characters and some of the control characters. Figure 73 shows a dump of the same screen, captured by a program called Image Capture, and dumped with Graphics loaded. For comparison, Figure 74 shows the same screen dumped to the same printer as text. The same graph, displayed on a graphics screen, appears in the Graphics entry.

Tips For related information, see ASCII Code, Echo, GrafTabl, Graphics, and Type.

```
 Apps  Disk  Create  Edit  Locate  Frames  Words  Numbers  Graph  Print 10 38 am
[Sales by Function] (G)

                              Sales by Function

                                                          Fees for Service
                                                          Royalties
                                                          Advances
                                                          Sales

                              1987 Sales Summary

@DrawGraph(salessum.C50:s‖ ry, 1987].Sales by Function ‖Frame:    2/3

  Screen resolution limits graph accuracy; printout will show full resolution
```

Figure 72: A text screen dump with graphics characters

```
 Apps  Disk  Create  Edit  Locate  Frames  Words  Numbers  Graph  Print 10 48 am
[Sales by Function] (G)

                              Sales by Function

                                                          Fees for Service
                                                          Royalties
                                                          Advances
                                                          Sales

                              1987 Sales Summary

@DrawGraph(salessum.C50:s‖ ry, 1987].Sales by Function ‖Frame:    2/3

  Screen resolution limits graph accuracy; printout will show full resolution
```

Figure 73: A text screen with graphics characters dumped as graphics

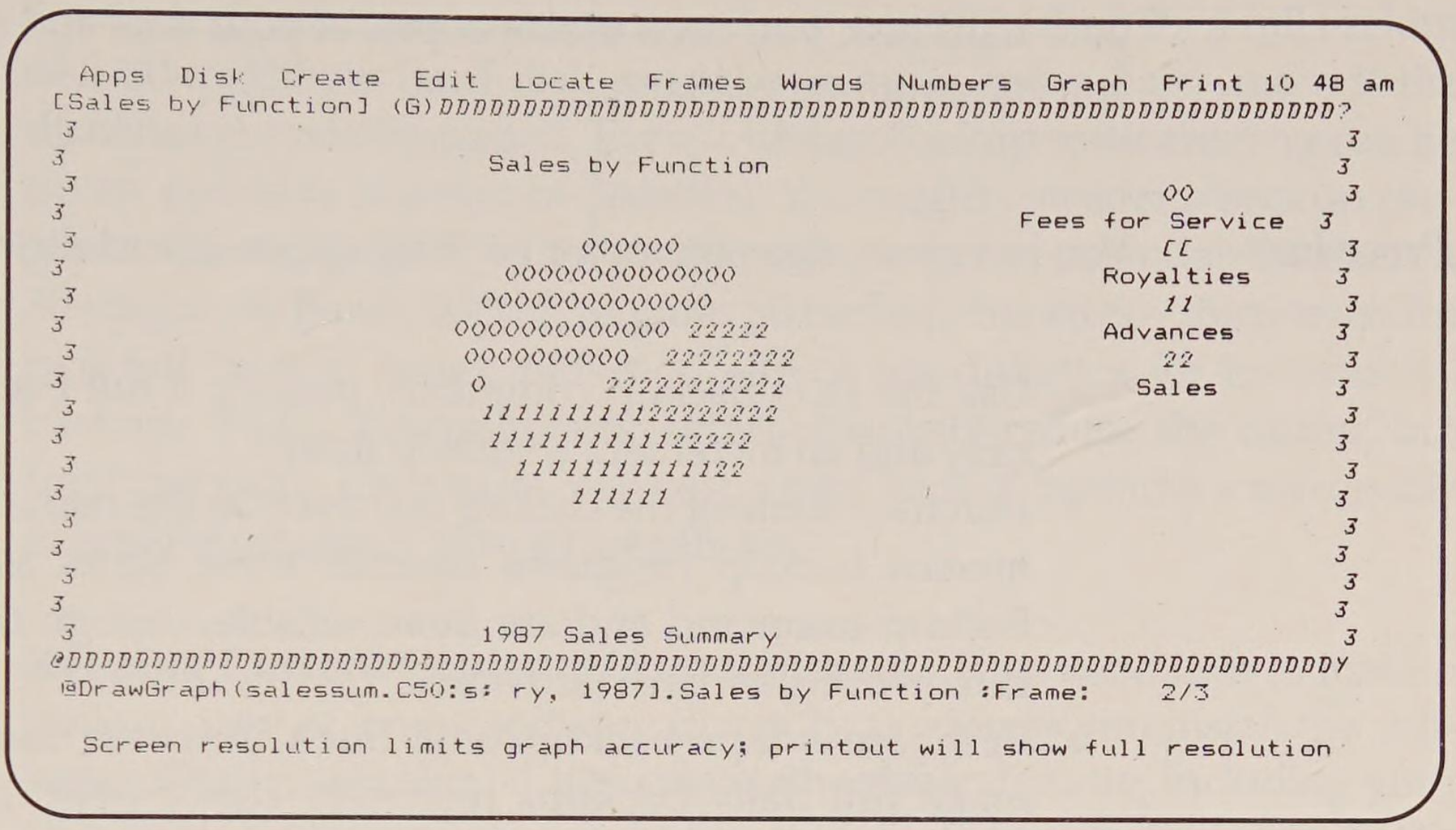


Figure 74: A text screen with graphics characters dumped as text

Hard Disks, Backing Up and Restoring

Overview

Hard disks are such a convenience that you're likely to take them for granted once you get used to them. You'll be tempted to dispense with diskettes altogether, except as a source of software to add to your hard disk. Thus, until you experience either a crash or a misbehaving program that wipes out massive amounts of data, you may be tempted to forget about backing up your hard disk. This temptation is compounded by the slowness and clumsiness of the DOS Backup command (see Backup).

But hard disks do fail, programs occasionally wipe out data, and it's easy to mistype a wildcard pattern and thereby wipe out a whole directory. Therefore, it's essential to make frequent and timely backups of the data on your hard disk.

There are two reasons for making backups: first, you want to be able to recover the data if your hard disk somehow gets trashed. For this purpose, a series of backup diskettes or a tape backup is adequate. However, if your computer becomes so severely disabled that you can't restore

your hard disk, you need usable copies of your data and software so that you can continue working.

Procedure You have four alternatives for backing up your hard disk:

- Use the DOS Backup command, making a full backup periodically and an incremental backup daily.
- Purchase some of the backup software on the market. Most commercial backup programs operate three times as fast as the Backup command and are more reliable, though they may not run on nonstandard hardware.
- Purchase a tape backup unit. Such units are fast enough to make full daily backups relatively convenient, but they are also expensive.
- Set up your directories so that they can be backed up easily with commands other than the Backup command.

When thinking about backups, you should consider whether you can use your backups directly, and how many eventualities you want to prepare for. Both these concerns are addressed in this entry.

The Backup command creates copies that cannot be used unless they are restored. If you use a DOS version earlier than 3.3, the copies can be restored only to a hard disk. You are still out of luck if your hard disk fails. The same is true for some commercial backup programs, and may be true for tape backups. Thus, you might prefer to use standard DOS commands to back up your data. Some strategies will be presented.

This brings us to our second consideration. Some commercial programs will fit 720K to 800K on a 360K diskette in a high density PC AT drive; however, you won't be able to read your backup diskettes if your PC AT fails and you are forced to work with a standard PC or a portable. To guard against a worst-case scenario, you should have all your current data on 360K diskettes in usable form as well as bootable diskettes of all your favorite software. On the other hand, if you expect to have access to a PC AT or equivalent, you can back up a lot of data to a 1.2-megabyte diskette.

Other considerations are time and cost. Using Backup for a full backup of a 30-megabyte hard disk can take an hour and a half—more if the diskettes are not formatted. Even with fast backup software, you can tie up an excessive number of diskettes. You might consider a backup program that writes 800K to 360K diskettes using a nonstandard format. A 30-megabyte hard disk still requires 80 backup diskettes—thirty each for two full backup series, plus two sets of ten diskettes for incremental backups. Full 1.2-megabyte diskettes obviously reduce the clutter, but they cost more. Even at its high cost, a tape backup becomes a reasonable alternative at about 30 to 40 megabytes.

Examples

Keeping Current Copies of Your Software It's a good idea to make a backup copy of a new software directory as soon as you install the software. Ideally, you should first create a bootable diskette, including your CONFIG.SYS file and any drivers that it loads, then use `copy . a: .` or the Xcopy command to copy the software directory to the diskette. If the software package is too big to fit on a single diskette, do an installation on diskettes as well as on your hard disk. Be sure to update your diskette copies when you change the software installation on your hard disk.

Keeping Current Copies of Your Data UPDATE.BAT, which follows, uses the Replace command to ensure that the copies of all files on the backup diskette are current. Then it uses a series of For loops to ensure that the files on the backup diskette match those on the hard disk directory (see For . . . In . . . Do).

If your data files use some extension other than .DAT, change the line that copies the data files to the diskette in drive A. If you have data files with several different extensions, you can either repeat this line with as many extensions as you need for each type of file or use replaceable parameters (see Batch Files) in the loop and enter the extensions on the command line:

```
for %%f in (*.%1 *.%2 *.%3 *.%4) do if not exist a:%%f copy %%f a:
```

To use the command in this form, invoke the program with a command line like this:

```
update txt fig dbf wks
```

On the other hand, if you're absolutely sure that there's nothing in the hard disk directory but data files, you can safely use *.*.

UPDATE.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`cls`	Clears the screen
`echo Deleting old .BAK files.`	Message
`del *.bak`	Deletes backup files
`echo Place backup disk in drive A`	
`pause`	Waits for a keypress
`replace c:*.* a: /d`	Replaces all files on the backup diskette that are newer than the copies on the backup diskette (/d switch is not available in PC-DOS or MS-DOS 3.3; DOS 4.0 substitutes a /u switch)
`a:`	Makes the backup diskette current
`for %%f in (*.*) do if not exist c:%%f del a:%%f`	Deletes any files from the backup diskette that don't also exist in the hard disk directory
`C:`	Makes the hard disk current
`echo on`	Restores display of commands
`for %%f in (*.DAT) do if not exist a:%%f copy %%f a:`	Backs up each file with a matching extension that exists on the hard disk but not on the diskette, displaying the name of each file as it is copied
`dir a: /p`	Presents a directory of the backup diskette

Backing Up Large Directories If you have DOS 3.2 or later, you can use the Attrib and Xcopy commands to make usable backups of complete directories (see Attrib and Xcopy). Unlike Backup, Xcopy will stop copying when the target disk becomes full. However, Xcopy can be set to copy only files whose archive bit is set (see Attrib and File Attributes for information about the archive bit). You can use the Attrib command to set the archive bit for all the files in the directory. Then (being sure you have some blank, formatted diskettes on hand), use the Xcopy command with the /m switch to copy the directory. Streamline the process with the program DIRBAK.BAT, which follows.

DIRBAK.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`cd 1`	Makes the directory named as a parameter current
`attrib +a *.*`	Sets the archive attribute for all files
`xcopy . a: /m`	Copies files with archive attribute set from the current directory to drive A, clearing the archive attribute after copying
`echo ^GInsert next backup diskette,` `echo or press Ctrl-Break to end.`	Message with a beep
`pause`	Waits for keystroke
`%0`	Calls DIRBAK.BAT to repeat the procedure

To use this program, simply enter the command `dirbak` followed by the name of the directory to back up. When your first backup disk becomes full, you will see the message

```
Insufficient disk space
nn File(s) copied
```

However, because the /m switch to Xcopy resets the archive bit after copying a file, simply insert another backup disk and press any key when you see the message `Strike a key when ready . . .` generated by the Pause command. The last line of the program calls the program itself and

repeats until you stop it. Only those files whose archive bit has not yet been reset will be copied.

When you see the messages

```
nn File(s) copied
Insert backup diskette.
Strike a key when ready . . .
```

and you do not see the message

```
Insufficient disk space
```

all the files in your directory have been copied. Press Ctrl-Break to end the program.

If your source directory has subdirectories and you want to maintain the subdirectory structure on the backups, you must modify the program to make it more specific. First, include lines to set the archive attribute of all files in all the subdirectories. This means that you'll have to include the subdirectory names in the program, with commands such as

```
attrib +a reports
attrib +a memos
attrib +a letters
```

or, if you use DOS 4.0, use the /s switch on the third line. Then you'll have to add the /s switch to the end of the fourth line.

Ensuring Timely Backups HDBACKUP.BAT, which follows, ensures that you do an incremental backup of your hard disk once a day, and a full backup on the last Friday of every month. Ideally, you would include the code in this program as part of your AUTOEXEC.BAT file.

HDBACKUP.BAT

COMMANDS	COMMENTS
`echo|more|date>todayis.bat`	The current date is recorded in TODAYIS.BAT.
`copy today current.bat`	The version of CURRENT.BAT used to place the current date in the environment is created.

continues

COMMANDS	COMMENTS
`call todayis`	TODAYIS.BAT is executed, placing the current date in the environment (see Redirection and Pipes for a complete explanation of the first three lines).
`cd\hdbackup`	\HDBACKUP directory is made current.
`call lastbkup`	LASTBKUP.BAT places the date of the last backup in the environment.
`friday`	FRIDAY.COM is executed; if it's the last Friday of the month, the batch file that does a full backup is executed. This program also creates LASTBKUP.BAT and creates the environment variable FB, which notes that a full backup has been done. For a full explanation of FRIDAY.COM, see ErrorLevel.
`if errorlevel 0 if not errorlevel 1 call fullback` `if %FB%!==%true! goto continue`	If a full backup has been done, the incremental backup call is skipped.
`if %today%@==%backup%@ goto backedup`	If today's date, placed in the environment by TODAYIS.BAT, is the same as the date of the last backup, placed in the environment by LASTBKUP.BAT, the hard disk has been backed up today. Execution bypasses the call to the incremental backup routine.
`call dailybak`	The incremental backup routine is called.

continues

COMMANDS	COMMENTS
`echo|more|date>lastbkup.bat`	LASTBKUP.BAT is updated, so that if you reboot the computer, you won't have to back up the hard disk again.
`goto continue`	If you did a backup, the message telling you you have already done so is skipped.
`:backedup` `echo The hard disk has been backed up today.`	
`:continue` `set fb=`	The environment variable FB is cleared.
`cd\`	The root directory becomes current.
`rem The rest of AUTOEXEC.BAT follows . . .`	

The program assumes that the files related to your backup procedures are in a separate directory called \HDBACKUP. This directory may include the Backup and Restore programs, your backup logs, or whatever software you use for the purpose. This directory also contains:

- A version of CURRENT.BAT, different from either of the ones shown in the Redirection and Pipes entry
- A batch file called LASTBKUP.BAT
- A batch file called DAILYBAK.BAT, which includes the commands you would enter to make an incremental backup
- A batch file called FULLBACK.BAT, which contains the commands to do a complete backup and ends with the lines

```
echo|more|date>lastbkup.bat
set fb=true
```

In addition, the program FRIDAY.COM, explained in the ErrorLevel entry, should be either in the \HDBACKUP directory or on the search path.

To fully understand the logic of the program, read the entry on Redirection and Pipes. Both the use of CURRENT.BAT and the technique of placing the date in the environment are explained there.

Before you run HDBACKUP.BAT, create LASTBKUP.BAT by entering the command

```
echo|more|date>lastbkup.bat
```

This file will be identical to the file TODAYIS.BAT, explained in the Redirection and Pipes entry. Remember, though, that TODAYIS.BAT is updated each time you boot your computer, but LASTBKUP.BAT is updated only when you back up your hard disk. (Note that most of the code for SETDATE.BAT, from the Redirection and Pipes entry, is also part of HDBACKUP.BAT. The routine that places the time in the environment is omitted.)

Tips For related information, see Backup, Batch Files, Replace, Restore, and Xcopy.

Hard Disks, Interleave Factor

Overview Disk sectors are logically numbered consecutively. However, on a hard disk, the logical numbering of the sectors often does not match their physical layout. That is, although two sectors with consecutive logical sector numbers are probably not next to each other on the disk, the hard disk controller persuades DOS to act as if they are.

There is a good reason for this. Although the read-write head can read disks quite rapidly, the central processing unit may not be able to process the information coming to it from the hard disk quickly enough to instruct the hard disk controller to read the next sector before the sector has already passed under the read-write head. If the sectors were numbered consecutively, the disk would have to complete a revolution before the next logical sector could be read.

In theory, therefore, hard disk sectors are numbered so that the sector with the next logical number passes under the head as soon as the CPU is ready to receive the information from it. In fact, the distance between logical sectors on your hard disk may not be optimal for your CPU's speed.

The number of physical sectors between two consecutive logical sectors is the *interleave factor*. As a rule the interleave factor for a hard disk is optimized at the factory for the computer in which the disk will be installed. Disks for XT-class computers have an interleave factor of 6, those for AT-class computers a factor of 3, and those for 386 PCs a factor of

1 (that is, no interleave factor). An interleave factor of 3 means that sector 3 is three sectors after sector 2, sector 4 is three sectors after sector 5, and so on. See Figure 75 for an illustration of this relationship.

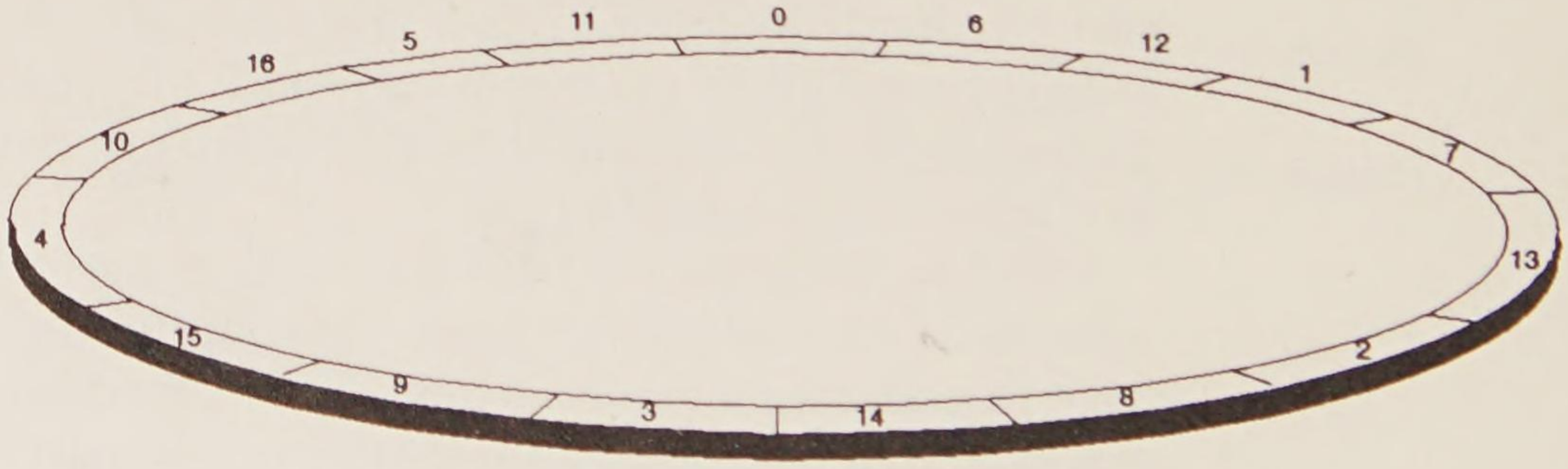
If you have a "turbo" computer—one that operates at a higher-than-standard clock speed (4.77MHz for PCs and PC XTs, 6 or 8MHz for PC ATs, and 16 or 20MHz for 386 PCs)—the disk's interleave factor may be too large. If so, your hard disk I/O may slow you down considerably.

Procedure

You can change the interleave factor of your hard disk by doing a low-level format. This is not the format done by the DOS Format command, but a formatting that prepares the disk surface. The problem is that you need to know what interleave factor you want to establish, and there's no easy way to do this. There are programs, however, to help you determine the interleave factor for your system's optimum performance. They include public-domain programs available from users' groups and electronic bulletin boards as well as several very good commercial utilities. You're reduced to trial and error without these programs.

Before you even think of doing a low-level format, **back up your hard disk completely!** (See Backup, and Hard Disks, Backing Up and Restoring.) A low-level format wipes out everything on the hard disk, far beyond the ability of any utility program to recover it.

To determine if a different interleave factor is better than your present one without utility software, first conduct some tests. Perform some operations that require extensive reading or writing to the disk. After the format, compare the time it takes to do the same operations with the



DOS logical sector numbers: Cylinder 0

Figure 75: Relationship of logical and physical sectors on a hard disk with an interleave factor of 3

times you recorded. If they take less time, your new interleave factor is more appropriate; if they take more, repeat the procedure with a different interleave factor.

You must use DEBUG to do a low-level format (see DEBUG). To begin, type `debug` to invoke the program. When you see the hyphen prompt, type `rax` and you should see

```
AX 0000
:
```

The colon is a prompt for entering information. Enter a four-digit number. The first two digits are the number of the hard disk (a disk named C is 00, a disk named D is 01, and so on). The second two digits specify the interleave factor. To format drive C with an interleave factor of 2, type

```
0002
```

At the next hyphen prompt, type

```
g=c800:5
```

Be careful about that 5—it may not be the right number. Whether this number is correct depends on your hard disk controller card. Check the manual that comes with your computer or your controller card. If the manual doesn't provide the necessary information, obtain the correct number from your disk controller manufacturer's technical support department.

You should see a series of prompts when you press Return. To proceed, type a `Y` for all of them. If you get cold feet, you can abort the process by pressing Ctrl-Break any time before it actually starts writing to your disk.

Low-level formatting takes anywhere from half an hour to several hours, depending on the size and design of your disk. Once the formatting is completed, you must create a DOS partition with FDisk, then perform a high-level format with the Format command (see FDisk and Format).

You can now restore some or all of your files and repeat your benchmark tests. If you have to repeat the process, you needn't back up the hard disk again.

Warnings

Under no circumstances should you do a low-level format unless you have first backed up your hard disk. This enterprise is never to be undertaken lightly.

Hard Disks, Structure of

Overview

The basic structure of a DOS-formatted disk is described under Diskettes, Structure of. However, hard disks differ from diskettes in important ways.

Hard disks consist of a series of *platters*—generally, at least two. As the number of platters increases, so does the amount of data a hard disk can hold.

Each platter has two sides for recording data. Platters are numbered beginning with 0, and each side of a platter has a separate number.

The hard disk is laid out as a series of concentric rings called *cylinders*, which are roughly equivalent to the tracks on a diskette (except that you might think of them as intersecting all the platters). The number of cylinders on a disk also affects the amount of data it can hold.

The first track on each platter is assigned a number, beginning with 0, before the second cylinder receives a number. Thus, on a three-platter disk, the outermost tracks are numbered 0 through 5, and track 6 will be physically next to track 0. Thus, when you format a hard disk, you will probably see a continuing display reporting something like the following:

```
Formatting side: 0  cylinder: 0
Formatting side: 1  cylinder: 0
Formatting side: 2  cylinder: 0
Formatting side: 3  cylinder: 0
Formatting side: 4  cylinder: 0
Formatting side: 5  cylinder: 0
Formatting side: 0  cylinder: 1
```

and so on. These messages usually replace each other on the screen, rather than scrolling. (In DOS 4.0, a message reports the percentage of the disk that has been formatted.) Each track is usually divided into seventeen *sectors* holding 512 bytes of data for standard hard disks. This is the layout expected by the computer's BIOS. Some special types of disks use other layouts. These disks also use a disk controller card peculiar to them, which re-maps the layout into a logical structure the BIOS can understand. Sometimes this procedure must be abetted by a device driver loaded through the CONFIG.SYS file (see CONFIG.SYS File).

All of track 0 on side 0 is reserved for the *partition table*, which tells DOS the following information:

- How much of the disk (and which part) you want to use for DOS
- How much and which part you want to use for other operating systems
- Which operating system is *active*; that is, which one will boot the disk

Note that this is track 0, side 0, beginning with sector 1 in the BIOS numbering system. In the DOS numbering system, this track is ignored altogether. For details on disk sector numbering, see Disk Sectors. For more information about the partition table, see FDisk.

The very first logical sector is the *boot record* (see Boot Record). This is a short program that starts the computer. A boot record appears at logical sector 0 of each logical drive that DOS can read. Two copies of the *file allocation table* follow immediately after it (see File Allocation Table). After these is the root directory (see Directories). The size of the file allocation tables and root directory varies with the capacity of the disk. The root directory of a hard disk normally is designed to hold up to 512 entries.

The rest of the disk is used for storing your data, and is divided into *clusters* (see Disk Clusters) of four or eight sectors, numbered consecutively beginning with 2. Hard disks of up to 16 megabytes generally have clusters of eight sectors each, and larger disks have four-sector clusters. However, you can accidentally cause a 10-megabyte disk to have sixteen-sector clusters. See Disk Clusters for details about how this can happen, and what to do if it does.

Before you can use a hard disk, it must be prepared in three separate ways. First, it must be *physically formatted*. This process divides the disk into sectors and gives each sector an address. For details, see Hard Disks, Interleave Factor.

Next, the disk must be *partitioned*. The FDisk program does this, creating the partition table, described previously.

Finally, it must be *logically formatted*. This is the function of the Format command. On a hard disk, this usually consists of simply creating a boot record, file allocation tables, and a root directory, or erasing the existing ones. However, some versions of DOS also do a read-write test to check the success of the format. If your DOS version does not, you may be able to recover data from any part of a reformatted hard disk except the root directory. If your DOS has a special

command that formats only the hard disk (perhaps called HDFMT), assume that it does the read-write test (see Format).

Tips

For related information, see Boot Record, Directories, Disk Clusters, Disk Sectors, Diskettes, Structure of, FDisk, File Allocation Table, and Format.

If

Syntax

{**1**} `if [not] string1==string2 command`
{**2**} `if [not] exist filename.ext command`
{**3**} `if [not] errorlevel n command`

Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

If is a batch command. You use it to determine whether a given condition exists before proceeding with the execution of the file. It allows you to select commands for execution or to redirect the flow of control to other parts of the batch file.

The If command can test for three different conditions:

- Equivalence between two strings
- The existence of a given file
- Exit codes passed to the ErrorLevel modifier (see ErrorLevel)

Include *not* in the If statement to test for the absence (rather than the presence) of a condition.

There is no equivalent to an Else command in the batch language. Therefore, you need two lines of code to choose between two actions:

```
if condition goto label1
goto label2
```

You may be able to avoid this construction by having the commands that would follow *label2* appear immediately after the If statement. See ACCOUNTS.BAT in the GoTo entry for a detailed example of this technique.

Procedure

Testing for Equivalence The If command tests for the equivalence of two strings by comparing a parameter or an environment variable to a string constant (see Set). The equivalence operator is a double equal sign.

To test a parameter, use a command of the form

```
if %1==string command
```

In this instance, if the first parameter on the command line matches *string*, DOS will execute the *command*; if not, it will not.

To test for the absence of a parameter, you must give DOS something equivalent to the absence of a parameter to look for. One way to do so is to ask it to check for a *null string*—a string of zero length. A null string is represented by a pair of quotation marks with nothing between them:

```
""
```

However, if your command takes the form

```
if %1=="". . .
```

DOS will look for a pair of quotation marks. You can either place the replaceable parameter in quotation marks

```
if "%1"=="". . .
```

or use a dummy string on each side of the equation:

```
if %1@==@ . . .
```

If there is no parameter, DOS will interpret the statement as

```
if @==@ . . .
```

which is equivalent. If there is a parameter (for example, TEXTFILE.ASC), DOS will interpret the parameters as

```
if TEXTFILE.ASC@==@ . . .
```

which is an acceptable statement, but is not an equivalence. DOS will then properly fail to execute the command in the If statement.

To test an environment variable, place the name of the environment variable between percent signs and precede the test string by a percent sign:

```
if %variable%==%string command
```

When DOS executes this statement, it looks in the environment space for the variable name, and substitutes the value assigned to it for *%variable%* (see Environment Space). Suppose the environment contained the variable

```
FLAG=true
```

If your batch file includes a statement beginning with

```
if %flag%==%true . . .
```

DOS will read the value of FLAG and interpret your statement as

```
if true==true . . .
```

Of course, the two values are the same, and the command in the statement would be executed.

However, with no environment variable FLAG, DOS translates this statement as

```
if ==true . . .
```

This statement is not true, and you would expect execution to proceed with the next line of the file. In DOS 2, it does; however, in DOS 3.0 or later, this line generates a `Syntax error` message. To avoid the error message, you can again use a dummy string on each side of the equivalence operator. Any character will do, as long as it's not a command character (see Command Characters):

```
if %flag%#==%true# . . .
```

If the environment variable does not exist, this statement is interpreted as

```
if #==true# . . .
```

This is, of course, not an equivalence, but it's not an error either. On the other hand, if the variable FLAG holds the value true, DOS will translate the statement to

```
if true#==true# . . .
```

Since both sides of the equation are the same, DOS executes the command.

Case Matching DOS pays attention to the case of the characters when checking a parameter for equivalence. Suppose, for example, that you want to pass a drive name to a batch file as a parameter. To check that parameter in an If statement, you must test for both uppercase and lowercase versions of the drive name.

Let's assume you want a batch file to run a program that looks for data files on any drive you specify on the command line. You might want to complete some special procedures if the files are on a RAM disk, but not if they are on a hard disk. You could include a command such as

```
if %1==d: goto ramdisk
```

However, if you invoke the program with a command line reading

```
PROGRAM D:
```

DOS interprets the If statement as

```
if D:==d: goto ramdisk
```

and will not go to the `:ramdisk` label.

To prevent this error, you must also include the line

```
if %1==D: goto ramdisk
```

Theoretically, you should test for every possible combination of uppercase and lowercase characters to avoid an error. This means that the longer the parameter is, the more tests you must perform. However, four tests are probably sufficient. If your If statement reads

```
if %3==all copy *.* %1
```

you can probably use just these four tests:

```
if %3==all copy *.* %1
if %3==ALL copy *.* %1
if %3==All copy *.* %1
if %3==aLL copy *.* %1
```

Even though other combinations are possible, these four are the most likely. The Caps Lock key is either on or off, accounting for the first two forms. The user may press the Shift key to capitalize the first letter. If Caps Lock is off, the third form would result. If Caps Lock is on, the fourth form would result.

When testing for an environment variable, you must match the case of the expected value. DOS interprets

```
TRUE==true
```

as false. However, you needn't match the case of the variable name. Although variable names always appear in the environment in uppercase letters, DOS recognizes references to them in either uppercase or lowercase.

Testing for the Presence of a File

{**2**} `if [not] exist` *`filename.ext command`*

There are times when you want to execute a command only if a given file is—or is not—present. For example, you might have a program that creates an error file if errors occur, but not otherwise. To print that error file, use the Exist modifier with a command such as

```
if exist errfile.txt copy errfile.txt prn
```

You can, of course, just use the command

```
copy errfile.txt prn
```

However, if there is no such file, you will see a

```
File not found
```

message. The `if exist` form prevents the error message.

Testing for Exit Codes

{3} `if [not] errorlevel` *n command*

Some programs, including some DOS external programs, generate an exit code when they end. This is a numeric code that tells whether the program worked properly, or, if it didn't, what went wrong.

You may want a batch file to take different actions depending on how a program ended. To use the exit code, set up a statement indicating exactly what action to take for each exit condition. This statement includes the If command, the ErrorLevel modifier, the specific numeric code, and the command to execute (see ErrorLevel).

Examples

Testing a Parameter The program MCOPY.BAT (shown in the Shift entry) illustrates the most common use of the If statement, which is redirecting the flow of execution. In this program, the line

```
if "%1"=="" goto end
```

tests to see whether there is a parameter on the command line. If there is no parameter to act on, you want the program to end. Therefore, execution is redirected to the label `:end`.

In one version of the program, the label is the last line; in another version (also in the Shift entry), a final command is executed after the `:end` label. This command is executed only once, after the rest of the file has completed its work.

Similarly, in LOGON.BAT in the Call entry, control is directed to a label if there are not at least three parameters. The program begins with the commands

```
if %1!==! goto message
if %2!==! goto message
if %3!==! goto message
```

The `:message` label demarcates a message explaining the proper syntax.

Nesting Ifs When testing for equivalence, you can simplify the code (and coincidentally increase the speed of execution) by using nested If statements. After executing a command in a batch file, DOS returns to the batch file to read the next line. If one line contains several commands, DOS won't need to read the file as often.

The program ACCOUNTS.BAT (shown in the GoTo entry) is set up to use the RAM disk unless you specify otherwise by entering a drive name. The program includes the line

```
if not %1@==C@ if not %1@==c@ goto ramdisk
```

This line tests for the parameter C, in both uppercase and lowercase forms. C is the drive name for the hard disk.

Notice also the use of *not*. If the command line includes no parameters, the commands to run the program on the RAM disk are executed. If the first parameter on the command line is something other than the name of drive C, the same thing will happen. Only if the C parameter appears on the command line will the code immediately following the If statement be executed.

Tips

For examples of programs that test environment variables, see Call; Hard Disks, Backing Up and Restoring; Redirection and Pipes; and Set. For examples of programs that test exit codes, see ErrorLevel.

Install=

Syntax

`install=filename.ext parameters`

DOS Versions

MS-DOS 4.0
IBM DOS 4.0

Overview

The Install= command is an alternative to AUTOEXEC.BAT commands for loading TSRs into your computer's memory (see AUTOEXEC.BAT File, TSRs). It is a configuration command, used in the CONFIG.SYS file (see CONFIG.SYS File). By installing programs of this type using the Install= command, you ensure that they are placed in low memory, which is a more efficient location for them than they normally occupy if you load them via the AUTOEXEC.BAT file or with DOS command lines.

The four DOS programs for which Install= is designed are

FASTOPEN.EXE
KEYB.COM

NSLFUNC.EXE
SHARE.EXE

Procedure

To use the Install= command, you enter a line in your CONFIG.SYS file. Each command must be entered flush with the left margin, on a line of its own; for example,

```
install=fastopen.exe c:=(50,25) d:=20
install=share.exe
```

If the program you are installing is not on the default directory of the boot disk, include the full path name:

```
install=c:\dos\share.exe
```

Although you can enter the equivalent commands on the command line without their extensions, you *must* include the file name extension in an Install= command. Otherwise, the syntax is the same as that which you would use to load the program via the command line.

Warnings

If you use a utility program to manage your TSRs, be aware that such a program cannot unload any TSRs loaded with the Install= command. Utilities of this type cannot control any programs loaded before them. Therefore, you should use the Install= command to load only TSRs that you want loaded in memory every time you use your computer, without exception.

Tips

You can get around this limitation by booting your system from a diskette with a CONFIG.SYS file that doesn't include the Install= commands.

You can load commercial TSRs, as well as DOS external programs, using Install=. I have, for example, loaded SuperKey with the command

```
install=c:\key\key.com
```

However, if you do, you should not, under any circumstances, unload these programs, even if they provide a method for doing so. Moreover, you cannot load programs that require their home directory to be current when you load them, because you cannot make any directory current until COMMAND.COM is in memory, and it doesn't get there until DOS is finished with CONFIG.SYS. Thus, you cannot load programs such as Side-Kick Plus and Referee with the Install= command.

Join

Syntax

{1} join *guestdrive* [*hostdrive*]*directory*
{2} join *guestdrive* /d
{3} join

DOS Versions

{**all**} MS-DOS 3.1 to 4.0
{**all**} PC-DOS 3.1 to 3.3
{**all**} IBM DOS 4.0

Overview

The Join command allows you to treat a disk drive as a subdirectory of another drive. The following conditions must be in effect for the Join command to work:

- The "host" subdirectory—the one to which the "guest" drive will be assigned—must be one level below the root.
- The host subdirectory must be empty.
- Neither the host subdirectory nor the guest drive—the one being added to the host disk—may be current.
- Neither the host subdirectory nor the guest drive may be part of a network.
- Neither the guest drive nor the drive containing the host subdirectory may be currently affected by an Assign or Subst command.

Procedure

Join is an external command; using it requires having the DOS program JOIN.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\join d: c:\data
```

You can join any type of drive that exists in your system—diskette drive, hard disk, or RAM disk—to a subdirectory of another drive.

When a join is in effect, you can no longer refer to the guest drive by its original name. Doing so results in the message

```
Invalid drive specification
```

Joining a Drive to Your Main Drive

{1} `join guestdrive [hostdrive]\directory`

To join a drive to a directory, enter the Join command, followed by the name of the guest drive, and the name of the host directory, separated by spaces. For example,

```
join d: \data
```

DOS will create the host subdirectory if it does not exist. If it exists but has files in it, DOS displays the message

```
Directory not empty
```

If the host drive is not current, preface the name of the host directory with its drive name:

```
join d: c:\data
```

When you enter this command, drive D and the directory C:\DATA will be the same. You can make drive D current by entering the command

```
cd\data
```

However, you can no longer refer to drive D in commands. Doing so results in the message

```
Invalid drive specification
```

Canceling a Join

{2} `join guestdrive /d`

To cancel the assignment made with a Join command, enter the command again, followed by the name of the guest drive and the /d (delete) switch:

```
join d: /d
```

You can also cancel a Join assignment by rebooting the computer.

Viewing the Joins in Effect

{3} `join`

Entering the Join command with no parameters produces a report of the Join assignments in effect. If you issue the Join commands described above and do not delete either assignment, this command will display

```
D: => C:\DATA
```

Examples

The environment imposes certain constraints if you use an environment such as Windows or DESQview. To begin with, these environments generally force you to keep their subdirectories current. DESQview, however, is quite reasonable about letting you run programs from batch files that can overcome its constraints. If you have a utility that you want to run from either the DESQview menu or from a .PIF file in Windows, you may find it difficult to run the program on any drive in your system unless you join the drive to your hard disk. To overcome this limitation, you might include a command such as

```
cd %1
```

in the batch file that calls the utility. This command will let you make any directory on your current drive current if you enter the name of the directory as a parameter to the batch file. If, however, you want to use the utility on another disk and have no opportunity to change drives, you can still access the other drives by using Join commands to make them into subdirectories. You will have to issue the Join commands before you start Windows, but you can both enter and cancel them in a DOS window in DESQview.

Warnings

Join is a dangerous command. When a join is in effect, the Assign, Backup, ChkDsk, DiskCopy, FDisk, Format, Restore, and Subst commands work erratically, unpredictably, or not at all.

Subst is also dangerous if one of the drives involved in a substitution is part of a network. At best, you may temporarily be unable to access a drive; at worst, you may either accidentally overwrite files in a different directory that have the same name or lock up your computer.

Tips

Except for the example given, the Join command is only marginally useful. The Subst command, which allows you to treat a subdirectory as though it were a separate drive, has many more applications (see Subst). This allows you to access subdirectories from programs that do not recognize path syntax (see Path Names).

Keyb

Syntax

{1} keyb *kc*[,*font*],[[*drive*][*path*\]keyboard.sys] [/e]
{2} keyb *kc*[,*font*],[[*drive*][*path*\]keyboard.sys] [/e] [/ID*nnn*]
{3} keyb /ID*nnn*
{4} install=keyb.com *kc*[,*font*],[[*drive*][*path*\]keyboard.sys] [/e] [/ID*nnn*]
{5} install=keyb.com /ID*nnn*

DOS Versions

{1} MS-DOS 3.3 to 3.31
{1} PC-DOS 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

The Keyb command remaps the keyboard to one of eighteen national keyboards other than the default. Keyb is incompatible with the Keyb*xx* command in DOS 3.0 to 3.2. The DOS 4.0 version also allows you to choose between two different keyboard layouts for three countries.

Procedure

Keyb is an external command; using it requires having the DOS program KEYB.COM on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\keyb sg
```

where sg is a two-character keyboard code.

You establish an alternative keyboard layout for your computer by entering the Keyb command at a DOS prompt or in your AUTOEXEC.BAT file.

You can load an alternative font at the same time as you load the keyboard layout by including the font number in the command:

```
keyb sg,850
```

The Keyb command reads the necessary information from the tables in KEYBOARD.SYS, the keyboard information file (see KEYBOARD.SYS). However, if KEYBOARD.SYS is in the root directory of the current drive, you needn't refer to it in the Keyb command. If it is not, you must include its location in your Keyb command so that DOS knows where to look for the information it needs.

If you use the 101-key enhanced keyboard (the keyboard with twelve function keys across the top), you should add the /e (enhanced) switch to the end of your Keyb command so that DOS knows where the keys are.

If you want to use the keyboard for the United States, English-speaking Canada, or Australia, ignore the Keyb command, since these countries use the default keyboard.

In DOS 4.0, you can load one of two alternative keyboard layouts for France, Italy, and the United Kingdom by using the /id switch and the appropriate number for the keyboard layout. You can also specify a national keyboard layout for any of the supported countries by using the /id switch and the appropriate number, instead of the keyboard code.

Once you load an alternative keyboard, you can switch back to the default (United States) layout by pressing Ctrl-Alt-F1. Similarly, you can switch from the default to the alternative layout by pressing Ctrl-Alt-F2. Table 17 shows the two-letter codes and the three-digit keyboard identifiers for the national keyboards for which information is included in KEYBOARD.SYS.

Examples To load the Belgian keyboard layout, use the command

```
keyb be
```

(be is the keyboard code for Belgium, as shown in Table 17.) If the KEYBOARD.SYS file is not in the root directory of the current drive, use a command such as

```
keyb be,850,c:\dos\keyboard.sys
```

This command assumes you also want to use the international font (font 850) to display the accented characters. If so, see both the lines to be placed in your AUTOEXEC.BAT file in the NLSFunc entry and the lines to be placed in your CONFIG.SYS file in the ChCP entry.

Warnings You may not be able to switch between layouts if your software takes over the Alt-F1, Ctrl-F1, Alt-F2, or Ctrl-F2 combinations. Consequently, you will need to experiment to determine whether you can toggle between layouts while your software is running.

Tips The alternative keyboards generally assign different symbols to the punctuation and other nonalphanumeric keys. However, you can enter the default characters for these keys by holding down the Alt key while pressing them.

TABLE 17

Keyboard Nationality Codes

COUNTRY	CODE	IDENTIFIER
Australia	US	103
Belgium	BE	120
Canada (English)	US	103
Canada (French)	CF	058
Denmark	DK	159
Finland	SU	153
France	FR	120, 189
Germany	GR	129
Italy	IT	141, 142
Latin America	LA	171
Netherlands	NL	143
Norway	NO	155
Portugal	PO	163
Spain	SP	172
Sweden	SV	153
Switzerland (French)	SF	150
Switzerland (German)	SG	000
United Kingdom	UK	168, 166
United States	US	103

Some of the keyboard layouts include *dead keys*—keys that do not generate a character when you press them. Instead, they place an accent mark over the next character you type (if the accented character is valid in the language). See your DOS manual for information on which keys are dead on which keyboard layout.

For related information, see NLSFunc, ChCP, Mode, and Keyb*xy*. The Mode entry includes a table of character set codes. The Country= entry includes a table of country codes.

The Keyb command has exit codes that you can test with the ErrorLevel modifier of the If command. See ErrorLevel for a table of exit codes.

KEYBOARD.SYS

Syntax

keyb *kc*[,*charset*],[[*drive*][*path*\]keyboard.sys]

DOS Versions

MS-DOS 3.3 to 4.0
PC-DOS 3.3
IBM DOS 4.0

Overview

KEYBOARD.SYS is the file that contains the information DOS uses to remap the keyboard for various national languages. Despite its name, KEYBOARD.SYS is not a device driver. However, DOS needs to be able to read it to switch from one keyboard layout to another when you install one with the Keyb command (see Keyb).

For a list of supported keyboard layouts and their two-letter codes, see the table in the Keyb entry.

Procedure

You establish an alternative keyboard layout for your computer by entering the Keyb command at a DOS prompt or in your AUTOEXEC.BAT file. This command reads the necessary information from the tables in KEYBOARD.SYS. However, if KEYBOARD.SYS is in the root directory of the current drive, you needn't refer to it in the Keyb command; if it is not, you must include its location in your Keyb command so that DOS knows where to look for the information it needs.

If you want to use the keyboard for the United States, English-speaking Canada, or Australia, ignore the Keyb command, since these countries use the default keyboard. For further information on keyboard layouts, see Keyb.

Examples

To use a Swedish keyboard when KEYBOARD.SYS is not in the root directory of the current drive, use a command such as

```
keyb sv,850,c:\dos\keyboard.sys
```

(SV is the keyboard code for Sweden.) This command assumes you also want to use the international character set (code 850) to display the accented characters. If so, see both the NLSFunc entry for the lines that

must appear in your AUTOEXEC.BAT file in the NLSFunc entry and the ChCP entry for the lines that must appear in your CONFIG.SYS file.

Tips

For related information, see NLSFunc, ChCP, Mode, and Keyb. The Mode entry includes a table of character set codes, the Country= entry includes a table of country codes, and the Keyb entry includes a table of national-keyboard codes.

Keyb*xx*

Syntax

keyb*xx*

DOS Versions

MS-DOS 3.0 to 3.2
PC-DOS 3.0 to 3.2

Overview

The Keyb*xx* command remaps the keyboard to a national keyboard other than the default. It does so by loading one of several files with names of the form KEYB*xx*.COM or KEYB*xx*.EXE, which appear on the MS-DOS system diskette. In contrast, the Keyb command in DOS 3.3 (and later) reads keyboard information for all nationalities other than the default from a single keyboard information file.

The more common releases of DOS 3.0 to 3.2 include the following keyboard-mapping files:

KEYBFR.COM
KEYBGR.COM
KEYBIT.COM
KEYBSP.COM
KEYBUK.COM

(Some versions use .EXE files instead of .COM files.) These files set up the keyboard for France, Germany, Italy, Spain, and the United Kingdom, respectively.

The default keyboard layout stays in memory. You can switch between the default and the alternative layout using the method described next.

Procedure

All of the keyboard-mapping files are external; using one requires having it on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\keybfr
```

Load only one alternative keyboard layout during a session. If you want to use an alternative layout routinely, place the Keyb*xx* command in your AUTOEXEC.BAT file.

Once you load an alternative layout, you can switch back to the default (United States) layout by pressing Ctrl-Alt-F1. Similarly, you can switch from the default to the alternative layout by pressing Ctrl-Alt-F2.

Warnings

Some versions of DOS use slightly different names for their keyboard-mapping files. For example, some use the form *xx*KBD.COM. Some versions also include more or fewer national keyboard files.

You may not be able to switch between layouts if your software takes over the Alt-F1, Ctrl-F1, Alt-F2, or Ctrl-F2 combinations. You will have to experiment to see whether you can toggle between layouts while your software is running.

Tips

The alternative keyboards generally assign different symbols to the punctuation and other nonalphanumeric keys. However, you can still use the default characters on these keys by holding down the Alt key while pressing them. For example, on a PC AT–style keyboard mapped for France, the <, > and \ characters are all produced using the upper left key on the keyboard, which shows a tilde and a grave accent on the American keyboard. At the same time, the key that shows the < and comma symbols produces the single quote (shifted) and the semicolon (unshifted). To produce a < character with the key that shows it on the keycap, hold down the Alt and Shift keys, and press the key. To produce a comma instead of a semicolon, hold down the Alt key and press the key.

Label

Syntax

{**1**} label [*drive*]
{**2**} label [*drive*]*diskname*

DOS Versions

{**all**} MS-DOS 3.0 to 4.0
{**all**} PC-DOS 3.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Label command is used to give a disk an internal volume label. This can be useful for identifying diskettes, especially if you have a series of related diskettes.

Procedure

Label is an external command; using it requires having the DOS program LABEL.COM (or LABEL.EXE) on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\label a:data #17
```

{**2**} label [*drive*]*diskname*

To enter or change a volume label, enter the Label command, followed by the name that you want to give the disk. If the disk is not the current one, preface the disk name with the drive name.

A volume label does not follow the format for a file name. It can be up to eleven characters in length, and cannot include a period. You can enter it in uppercase or lowercase, but it will always appear in uppercase unless you use a commercial utility program to enter lowercase characters.

You can enter spaces as part of a volume label. The other characters you can enter depends on your particular version of DOS. Some permit any characters that are valid in file names; others accept only alphanumeric characters. If you have entered characters that your DOS finds unacceptable, you will see the prompt

```
Invalid characters in volume label
Volume label (11 characters, ENTER for none)?
```

Enter a new label, with something else in place of the invalid characters. If you're not sure which characters are acceptable, omit any characters that are invalid in file names.

{**1**} label [*drive*]

You can enter the command without including a disk label. If you do so, you will see a prompt of one of the following forms:

```
Volume in drive has no label
Volume label (11 characters, ENTER for none)?
```

or

```
Volume in drive C is HARD DISK C
Volume label (11 characters, ENTER for none)?
```

You can then enter the new volume label.

To remove a volume label, enter the Label command without a disk label, and press Return at the prompt.

If a volume label exists, it always appears as the first line of a directory display. If the disk has no label, the first line of the directory display is

```
Volume in drive d has no label
```

To see a volume label without displaying a directory, use the Vol command (see Vol).

Warnings

If the disk already has a label and you don't want to change it, press Ctrl-Break at the prompt rather than Return.

Tips

You can also give a disk a label by using the /v switch to the Format command (see Format). This is the only way to give a disk a volume label in DOS 2. You cannot change volume labels using DOS 2.

Some releases of DOS ask you to enter the volume label of the target disk when you try to format a hard disk that is already formatted. The purpose of this prompt is to confirm that you really mean to format it. If

you formatted the disk with a version of DOS other than the one you are now using, DOS may reject some characters in your current volume label as invalid and refuse to proceed. This may also happen if you have forced some lowercase characters into the volume label.

To get past this point, enter the Label command without a disk name, then press Return at the prompt to remove the current volume label. You can then press Return when the Format command asks for the volume label, and proceed as usual.

LastDrive=

Syntax

`lastdrive=`*d*

DOS versions

MS-DOS 3.0 to 4.0
PC-DOS 3.0 to 3.3
IBM DOS 4.0

Overview

LastDrive= is a configuration command which determines the maximum number of virtual drives you can have in your system. Unless you use this command to allow for additional drives, DOS allows access only to drives A through E, or, if you have a network loaded, A through N.

The LastDrive= command does not limit access to physical drives, or drives such as RAM disks installed by device drivers. Assuming A and B are reserved for diskette drives, you can have more than five such drives and experience no problems.

Procedure

LastDrive= is a configuration command, which is entered via the CONFIG.SYS file. Normally DOS recognizes only drive names through E, unless you have physical drives with higher designations in your system or have a network loaded (on a network drive names through N are recognized).

If you have already used all of the allowable drive names, and you try to create a network virtual drive, or to create a virtual drive using the

Subst command, you will see the message `Invalid drive specification`. To prevent this, add a command to your CONFIG.SYS of the form

```
lastdrive=d
```

where *d* is a letter after E, or after N if you use a network. If you find the Subst command convenient, it costs you nothing to set LastDrive to Z.

The new LastDrive designation does not become active until you reboot your computer after adding the command to your CONFIG.SYS file.

Tips

If you need to use the Subst command, and you don't have any drive names available, you can use the name of an existing drive. If you do, however, you will lose access to the physical drive of the same name until you undo the substitution.

Link

Syntax

{1} `link`
{2} `link objectcode[,runfile],[listfile],[libraries][/switches]`
{3} `link @responsefile`

DOS Versions

{1}{2} MS-DOS 2.1
{1}{2} PC-DOS 1.0 to 2.1
{all} MS-DOS 3.0 to 4.0
{all} PC-DOS 3.0 to 3.2

Overview

Link is a program that turns assembled code created with an assembler or compiler into executable programs. It also combines independent object-code modules created by a high-level language compiler into a single program.

Although assemblers and compilers create code that the computer can understand, it must be linked into executable (.EXE) files before it can be run. The linker performs this function.

If you program in assembly language or in a high-level language such as C, Pascal, Turbo BASIC, or QuickBASIC, you must use a linker to turn your object code modules into executable code. (You do not need to link

programs in interpreted languages such as BASICA and GWBASIC.) In all probability, a linker was packaged with your language software. If it was not, you can use the linker packaged with DOS. Generally, you should follow your language manual regarding options and switches to use when linking a program.

Table 18 shows the switches available with Link. You may enter them either as their one-letter abbreviations or in full. Some versions of Link have additional switches. If your version does, the abbreviations you use must be long enough to distinguish the switch you want to use from all other available switches. For example, some versions include the switches /noignorecase, /nodefaultlibrarysearch, and /nogroupassociation. To abbreviate these switches, use /noi, /nod, and /nog, respectively.

Usage

Link is an external command; using it requires having the DOS program LINK.EXE on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:link a:testprog
```

TABLE 18 **Linker Switches and Their Effects**

ABBREVIATON	NAME	FUNCTION
/d	/dsallocate	Loads data at the high end of the data segment (you must use this switch with Pascal and FORTRAN programs)
/h	/high	Places the run file as close to the top of memory as possible (you must not use this switch with Pascal and FORTRAN programs)
/l	/linenumbers	Places line numbers in the list file (you may find this helpful when debugging)
/m	/map	Lists all global symbols, with their definitions, to a file or the printer
/o	/object	Links Pascal and FORTRAN oject modules
/p	/pause	Interrupts linking until you press Return
	/stack:*nn*	Sets a fixed stack size of *nn* in the executable file
	/x:*nn*	Sets the total number of segments to *nn* in the executable file

If you use DOS 3.0 or later and the program is neither in the current directory or on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\link infields+display+maincode, newprog, nul.lst
```

You may precede the name of any file used in the process with a drive and/or path name if it is not in the current directory.

```
{1} link
```

You can enter the Link command with no parameters. DOS will display the following prompts, in turn:

```
Object modules [.OBJ]:
Run File (OBJECTFILE.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
```

You must enter the names of the object modules you want to link in your final program, separated by plus signs. You needn't include the .OBJ extension. If you have too many names to fit on one line, end the line with a plus sign. The prompt will reappear, and you can enter the names of more object modules.

If you enter only one object module, its name becomes the name for the run file. If you enter more than one object module, the name of the first module becomes the name of the run file *OBJECTFILE*. You can enter a different name for the program at that point if you want to. To accept the default, simply press Return at the prompt.

The list file is a cross-reference of internal linkages. No list file is created unless you use the /m or /l switch, or offer a file name at this point. The name NUL.MAP in the prompt stands for no file. List files have the extension .MAP unless you specify a different one. You can enter PRN if you want the linkage map to go to the printer.

Most high-level languages include a default library. If yours does not, you must either enter the name of your default library or, before you invoke the linker, create the environment variable LIB, with a command such as

```
set lib=mylib.lib
```

(See Environment Space and Set for an explanation.) To use more than one library, enter the names of the libraries separated by plus signs. You needn't enter the .LIB extension. If you end the list with a plus sign, the

prompt will reappear, and you can enter more library names. To accept the default, press Return.

To enter switches, enter them at the end of the response to the appropriate prompt.

{2} `link objectcode[,runfile],[listfile],[libraries][/switches]`

You can also link a program by entering all the options on the command line. For example,

```
link infields+display+maincode,newprog,,mylib.lib+spec.lib
```

This command also links the modules INFIELDS.OBJ, DISPLAY.OBJ, and MAINCODE.OBJ. Link does not create a list (.MAP) file because nothing appears between the two commas that would normally surround its name. The linker tries to resolve all references to external modules not included on the command line by searching the library files MYLIB.LIB and SPEC.LIB. Assuming no errors, the result will be an executable program called NEWPROG.EXE.

Alternatively, you can simply enter

```
link infields+display+maincode
```

and let the linker prompt you for the other file names.

{3} `link @responsefile`

DOS 3.0 and later allows you to link a program by creating an ASCII text file containing the responses to all of the linker's prompts. The responses to each prompt should be on a single line. Then, when you invoke the linker, follow the command with an @ symbol and the name of the file containing the responses to the prompts. To continue the same example, you can create the response file NEWPROG.TXT, containing the lines

```
infields+display+maincode
newprog
mylib.lib+spec.lib
```

You can then invoke the linker with the command

```
link @newprog.txt
```

Tips

Consult your language manual to guarantee that you are linking your program correctly.

LPT*n*

Overview

LPT*n* is the name for the line printer, specifically, a printer attached to a parallel port (see Ports). There can be up to three parallel devices named LPT1, LPT2, and LPT3 attached to a DOS computer. (See Device Names.)

MD

Syntax

{**1**} `md` *`path`*
`md`*`path`*
{**2**} `md` *`drive path`*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The MD command creates a new subdirectory on a disk with tree-structured directories. Subdirectories are an essential element of hard-disk organization and may also be useful on high-capacity diskettes. MD stands for Make Directory. The longer form, MkDir, is equivalent.

You will want to create a new directory under the following circumstances:

- Whenever you add a new software package to your hard disk
- When the root directory is full
- Whenever a directory has so many file names that you can't find what you want
- When you want to change the logical organization of your files

Virtually every large-scale software package includes instructions suggesting that you create a separate directory for it. This is a wise idea since today's software may include dozens of files. If you don't create a separate directory, you may be unable to find the files that belong together. Also, note that many packages have files with the same names (such as

INSTALL.EXE and READ.ME). You cannot have two such files on the same disk in a single directory, but you can have them on the same disk in separate subdirectories.

The root directory of a disk holds a finite number of file names (normally 512). When you reach the limit, it is imperative that you create subdirectories if you want to add more files to the disk. Subdirectory names must appear as entries in a directory—in the root directory, if you do not already have other subdirectories. You will therefore need to erase some files before you can create a subdirectory.

Your directory tree can save you time if it helps you find the files you need; if it does not, it can impair your productivity. Directories that are not logically organized or that are too full impede your work. Thus, you may need to make new subdirectories periodically to make your tree more efficient. There is no set rule regarding the best organization—you must set up a structure with which you can work comfortably (see Directories).

You may also want to create subdirectories on diskettes, especially if you use high-capacity diskettes. Subdirectories are useful for keeping groups of unrelated files on the same diskette for backup.

Procedure

In the discussion that follows, assume that the hard disk is drive C. The directory structure will follow Figure 28 in the Directories entry. For the following examples, you may want to use Figure 28 for reference.

{**1**} `md` *`path`*
`md`*`path`*

To create a new subdirectory on the current drive, enter the MD command followed by the name of the directory you want to create. To create a subdirectory that is a child of another subdirectory, you must specify the path to the directory you want to create.

Like all commands using path names, MD can use either absolute or relative path names. When using an absolute path name, you need not enter a space between the command and the path name.

{**2**} `md` *`drive path`*

To create a new directory on a disk that is not current, include the drive name in the command before the path name.

Examples

Suppose it's September 1, and you need to begin keeping records for the fourth quarter of the year. Therefore, your regional directories for reports and invoices need a new subdirectory for the new quarter. If your current directory is anywhere on drive C, you can issue these commands:

```
md\reports\east\qtr4
md\reports\midwest\qtr4
md\reports\south\qtr4
md\reports\west\qtr4
md\invoices\east\qtr4
md\invoices\midwest\qtr4
md\invoices\south\qtr4
md\invoices\west\qtr4
```

Alternatively, you can first make \REPORTS current, then issue shorter commands such as `md east\qtr4` and repeat the procedure after making \INVOICES current.

When installing a new software package, you may need to change to drive A and create a new subdirectory on the hard disk. To accomplish this, use a command of the form

```
md c:\software
```

Warnings

Enter a space between the command and the parameter when an MD command includes a drive name. Do not enter a space between the drive name and the path name. The following forms would result in error messages:

```
mdc:\reports\midwest\qtr4
md c: \reports\midwest\qtr4
```

Tips

When installing new software, you will probably need to make the new package's directory current with a command such as

```
cd c:\software
```

as soon as you have created the new directory. You can save some keystrokes by entering the first command, and then typing a `C` and pressing

the DOS command key F3. This key will enter all the remaining characters from the previous command in their equivalent positions on the command line.

For related information, see CD, RD, Path Names, and Directories.

Mem

Syntax

{1} mem
{2} mem /program
{3} mem /debug

DOS Versions

{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

The Mem command gives you a summary report on the memory installed in your computer, and how much of it is currently available for your use. Optionally, it will report the names of programs currently installed in your computer's memory.

Procedure/ Examples

Mem is an external command. Therefore to use it, you must have the DOS program MEM.COM on a disk in your computer. If the program is not in the current directory or on the search path, preface the command with a drive name and/or path name; for example:

```
c:\dos\mem
```

{1} mem

To get a brief report of the amount of memory in your computer, and the amount currently available, enter the Mem command with no parameters. The report, which will resemble Figure 76, shows you the total amount of conventional memory, extended memory, and expanded memory conforming to the Lotus-Intel-Microsoft Expanded Memory Standard, revision 4.0 (LIM-EMS), and the amount of each that is is use (see Memory). It will not report on expanded memory that does not conform to this standard. The report is similar to the RAM report included

```
 655360 bytes total memory
 655360 bytes available
 481632 largest executable program size

3014656 bytes total EMS memory
 131072 bytes free EMS memory

 131072 bytes total extended memory
 131072 bytes available extended memory
```

Figure 76: The Mem report

in the output of the ChkDsk command, but it doesn't require reading a disk, so you get the report much more quickly than you do with ChkDsk (see ChkDsk). The "largest executable program size" in the Mem report is equivalent to the number of "bytes free" reported by ChkDsk.

{2} `mem /program`

Adding the /program parameter to the Mem command generates a four-column report, showing:

- The load address of each item in memory, expressed as a hexadecimal number
- The name of the program, where appropriate
- The size of the program in bytes, expressed as a hexadecimal number
- What type of item it is

As you can see from Figure 77, areas used exclusively by DOS are reported, as well as all the areas reserved by the CONFIG.SYS file. Note that the report includes all the areas set up by DOS, whether or not they have been established specifically by the CONFIG.SYS file. The CONFIG.SYS file in the computer from which this report was generated included no FCBs= or Stacks= statements; nonetheless space is reserved for a default number of each. Note also that an environment is created for each program, and that portions of the system programs IBMDOS.COM and COMMAND.COM may be loaded more than once. You can surmise from the presence of VDisk among the devices in low memory that the large block of occupied EMS memory is accounted for by a RAM disk (see VDISK.SYS, RAMDRIVE.SYS).

```
C:\>mem /program

  Address      Name          Size       Type
  _______      _______       _______    _______

  000000                     000400     Interrupt Vector
  000400                     000100     ROM Communication Area
  000500                     000200     DOS Communication Area

  000700       IBMBIO        0025A0     System Program

  002CA0       IBMDOS        008900     System Program

  00B5A0       IBMBIO        00BDC0     System Data
                 EMM         0029C0      DEVICE=
                 EDISK       000550      DEVICE=
                 DM          000AE0      DEVICE=
                 MSMOUSE     001A20      DEVICE=
                             0004B0      FILES=
                             000100      FCBS=
                             005330      BUFFERS=
                             000370      LASTDRIVE=
                             000CD0      STACKS=
  017370       COMMAND       001640     Program
  0189C0       IBMDOS        000030     -- Free --
  018A00       COMMAND       0003C0     Environment
  018DD0       IBMDOS        000040     -- Free --
  018E20       APPEND        001E20     Program
  01AC50       KEY           000090     Environment
  01ACF0       FASTOPEN      001A60     Program
  01C760       KEY           00DE80     Program
  02A5F0       MEM           000090     Environment
  02A690       MEM           012F60     Program
  03D600       IBMDOS        0629F0     -- Free --

    655360 bytes total memory
    655360 bytes available
    481632 largest executable program size

   3014656 bytes total EMS memory
    131072 bytes free EMS memory

    131072 bytes total extended memory
    131072 bytes available extended memory
```

Figure 77: The Mem/Program report

{3} mem /debug

Adding the /debug parameter to the Mem command produces a bit of additional detail. Specifically, it adds a list of the internal DOS device drivers, and some additional information about the contents of EMS memory.

Tips

See Memory for related information.

Memory

Overview

Read-Only Memory Your computer has several types of memory, and you may be able to add other types. First, there is read-only memory (ROM). Several ROM chips on your computer's motherboard store the Basic Input/Output System (BIOS), which controls the interactions between the parts of your system at the most primitive level. Details of the ROM BIOS are beyond the scope of this book. If your computer is made by IBM, you have additional ROM that contains a portion of the BASIC programming language. If you turn on your computer without a boot disk, it will come up in this abbreviated version of BASIC. If you have a hard disk or an EGA or VGA monitor, there are ROM (and RAM) chips on its controller card.

Conventional Memory More familiar to you is random-access memory (RAM). This is the memory where you load your programs, where the programs process their files, and where DOS commands are executed. Indeed, nothing happens in the computer until some information is placed in memory.

You generally think of your system as having a specific amount of memory installed (for example, 256K, 512K, or 640K). But not all memory is created equal. This RAM—which we may refer to as *base* or *conventional memory*—is only part of the memory in your computer.

In addition to base memory, your computer also has *system memory*. To understand the relationship between base and system memory, you need to know something about the way DOS addresses memory.

Memory Addressing The RAM in your computer is addressed in *segments* of 64K (1000 hex) each. Each segment is designated by a single-digit hexadecimal number. Thus, there are sixteen memory segments, designated 0 through F. Conventional memory occupies up to the first ten of these segments, 0 through 9. Therefore, unless you have one of several unusual memory boards, the maximum conventional memory you can have is 640K—ten blocks of 64K.

Each byte is designated by a four-digit *segment* address (starting with the segment designation), followed by a four-digit *offset* address. The latter is the number of bytes (again in hex) from the beginning of the segment. Thus, the 256th byte of the second segment, speaking decimally, would have the hexadecimal address of

```
1000:0100
```

Both the segment and the offset addresses are numbered beginning with 0. If you've ever used DEBUG, you will recognize this address form (see DEBUG).

Figure 78 shows a diagram of base and system memory for the PC family of computers. Note that the memory available to you is divided into ten 64K blocks. You can have any number of them filled provided

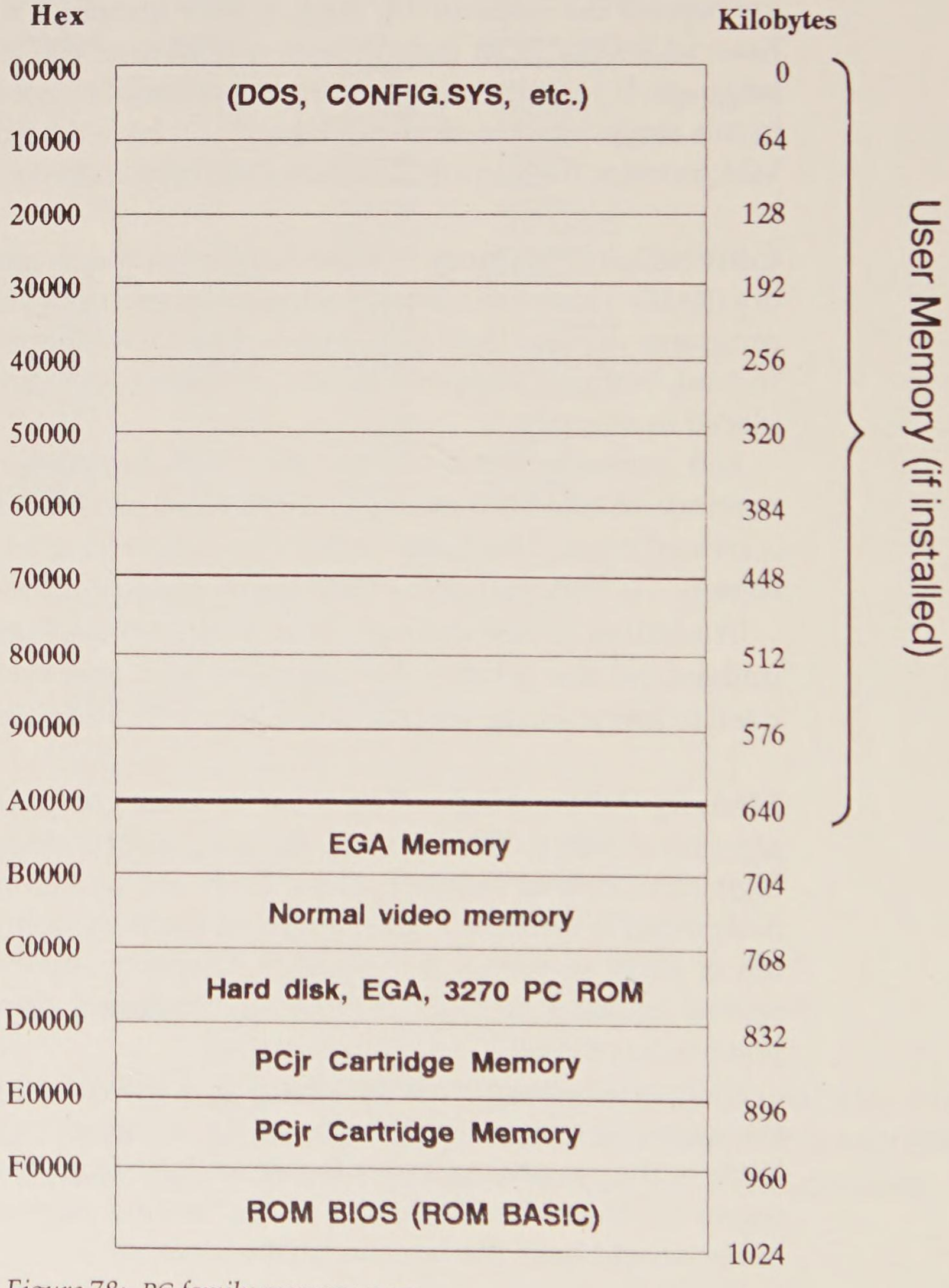


Figure 78: PC family memory map

they are filled consecutively beginning with address 0. If some blocks are empty, addresses are not changed in system RAM (shaded in the diagram).

System Memory You may have noticed that sixteen segments of 64K add up to 1 megabyte, not 640K, and wondered what the remaining 384K is used for. Actually, the existence of this 384K is somewhat of an illusion. There is an additional 384K of *address space*, but there may not be any memory installed at some of these addresses. These addresses are used, among other things, to provide an address location for the various types of ROM. As Figure 78 shows, the very last segment is where the ROM BIOS, and ROM BASIC, if you have it, are addressed. Other address segments are used for expansions to the ROM created for use with hard disks, enhanced graphics adapters, and 3270 PCs, and located on boards installed in the expansion slots.

Other addresses are assigned to video memory, which retains a map of the screen. In text mode, the map includes space to record one character for each of the 2000 characters that can fit on the normal 25 × 80 screen and its attribute (underlined, blinking, color, etc.). In graphics mode, the map includes information for every dot on the screen—whether it is on or off, and, if you have a color system, what color it contains if it is on. Some video memory may be installed on the video card itself. Some of the system memory is reserved for PC*jr* cartridges.

Adding Memory Notice in Figure 78 that standard video memory starts at B0000, and that A0000 is reserved for extended video memory. If you don't need extended video memory, can you reclaim this 64K block as base memory? The answer is a qualified yes. If you use a monochrome (not monographics) or color graphics adapter, your system is not using the A0000 block. You can add it to your system memory, using a public-domain program called MEM704.COM, if you can find the program and if you already have 640K of base memory.

Depending on your hardware, you may be able to increase your base memory to 640K by adding chips to your motherboard or by adding a memory card. (Early model IBM PCs had only 64K on the motherboard, with no room for more. Later model PCs had 256K, also with no room for more. Most clones have room for 512K or 640K.) More on this point follows.

Types of Additional Memory You can add two types of memory to a PC: conventional and *expanded*. You can also add *extended* memory to a

computer with an 80206 processor (that is, a PC AT or compatible) or a 386 PC. If you have no room on your motherboard for additional memory chips, add memory by installing memory cards in the expansion slots in your computer. There are memory cards that simply fill your base memory to 640K. Many of these cards also include such facilities as a real-time clock/calendar, additional ports, and software to create RAM disks, disk caches, or print spoolers (see Ports, Serial and Parallel).

If you need more than 640K, you can add an *expanded* memory board. This is memory you install above the top of the 1-megabyte memory range that DOS can address. This type of memory requires you to load a special expanded memory driver in your CONFIG.SYS file with the Device= command (see CONFIG.SYS File and Device=). It is probably called something like EMM.SYS. This driver should conform to a set of specifications developed by Lotus, Intel, and Microsoft, known as the LIM-EMS standard. Its original version had the release number 3.2. It contains a system for placing portions of data in expanded memory, and giving you access to it via a "window" in conventional memory. This happens invisibly. It manages this trick by using some of the address space between 640K and 1 megabyte, which DOS can address, into which it maps areas of the expanded memory.

Do not think of this memory as simply additional conventional memory. If you have 640K of base memory and you load device drivers and TSRs taking 300K, you will still only have 340K of base memory, no matter how much expanded memory you install (see TSRs). To take advantage of expanded memory, your software must be designed to use it. (However, some of the newer TSRs can store part of their code in expanded memory.) Software that uses expanded memory does so by storing pointers in base memory to the data in expanded memory, in order to find it later. Thus, even though your software may theoretically be able to address 8 megabytes of expanded memory, it will probably run out of space to store the pointers before you use anywhere near that much.

Most expanded memory boards can be configured to fill your base memory to 640K, if you don't already have that much base memory. If you use them for this purpose, you will have less expanded memory.

Even if your software doesn't take advantage of expanded memory, you can still use some for RAM disks, print spoolers, disk caches, and other software that creates buffers in memory. (See VDISK.SYS,

RAMDRIVE.SYS, and AUTOEXEC.BAT File). The advantage is that even though software that creates these buffers takes up conventional memory as a TSR, the buffers themselves can reside in expanded memory. This leaves more base memory for your applications. These various types of buffers allow you a limited degree of multitasking and also speed up your system by reducing the time it takes to read data from disks.

In addition to the LIM-EMS specification, a second standard, called *enhanced expanded* memory, was developed by AST Research, Quadram, and Ashton-Tate. This EEMS standard added capabilities to the LIM standard.

There are two principal differences. First, the LIM standard allows you to view expanded memory in up to four 16K chunks, whereas the EEMS standard uses up to four 64K chunks. (The latest version of the LIM-EMS standard release, 4.0, has adopted this larger "memory page size.") Second, extra hardware on EEMS boards allows you to run several programs at once by placing program code as well as data in expanded memory. This requires, however, that you have a *control program* such as DESQview, or Concurrent DOS/XM, or Microsoft Windows to manage the multitasking for you.

Memory boards that conform to the LIM-EMS 4.0 specification have extra hardware similar to that on EEMS boards. Thus, although you can probably get a LIM 4.0 driver for a board designed to the LIM 3.2 specification, you should not expect to do multitasking with such a board.

If you want to use enhanced expanded memory for multitasking, you can do so most efficiently by reducing the base memory on the motherboard to the lowest figure your computer can accept. You can then fill the remainder to 640K with memory from the enhanced expanded memory board. This allows the sophisticated EEMS or LIM 4.0 manager to manage some of your base memory as well.

As noted, if your computer has an 80286 or 80386 processor you can have extended memory. Extended memory is the same as conventional-memory as far as the microprocessor is concerned. However, DOS can address only 640K of conventional memory, under any circumstances. Therefore, DOS cannot know about any additional memory other than its system memory.

Extended memory is addressed beginning at 1 megabyte, just as expanded memory is. However, if you have an operating system that

can address it, you can use extended memory as though it were contiguous with the base memory. OS/2, Concurrent DOS/XM, Concurrent DOS/386, and UNIX can all address extended memory directly, even if DOS cannot.

Unless you use one of these alternative operating systems, about the only thing you can do with extended memory is to place a RAM disk in it. See VDISK.SYS and RAMDRIVE.SYS for details.

There is one small exception, however. A few programs—AutoCAD, the R:BASE and Framework programs among them—can use extended memory as a data-swapping area, as though it were expanded memory, if you install the correct driver in the program.

Memory Speed As a rule, you use extra memory to perform larger tasks and to speed up your system. However, if your added memory boards are not appropriate for your system, they can actually slow your system down. On an 8088-based computer, the *memory bus*—the channel through which data is read into and from memory—is 8 bits wide. That is, it can move 8 bits (one byte) at a time. In contrast, the memory bus on 80286-based computers is 16 bits wide, and some of the memory in 80386-based computers uses a 32-bit bus. Thus, it's important to match the memory board correctly to the type of computer you have.

In addition, the memory chips themselves operate at various speeds, measured in nanoseconds—billionths of a second. If your computer has a "turbo" speed, you will need faster chips.

Tips

DOS 4.0 includes some limited support for expanded and extended memory in the form of the device drivers XMAEM.SYS and XMA2EMS.SYS, which work only with specific memory boards manufactured by IBM. See XMAEM.SYS and XMA2EMS.SYS for more information.

MkDir. *See* MD

An alternative name for MD.

Mode

Syntax

{1} mode comn:*bps*,*parity*,*databits*,*stopbits*

{2} mode comn:[baud=]*bps*,[parity=]*parity*,[data=]*databits*,
[stop=]*stopbits*

{3} mode comn:*bps*,*parity*,*databits*,*stopbits*[,p]
mode lptn:=comn:

{4} mode comn:[baud=]*bps*,[parity=]*parity*,[data=]*databits*,
[stop=]*stopbits*[,[retry=]*c*]
mode lptn:=comn:

{5} mode lptn:[*chars*][,[*lines*][,p]]

{6} mode lptn:[cols=*chars*][,[lines=*lines*][,retry=*c*]

{7} mode *display*[,t]

{8} mode con [*display*] [[cols=]*chars*][,[[lines=]*lines*]

{9} mode con c[ode]p[age] prep[are]=((*charsets*)[*drive*]
[*path*\][*display*.cpi)

{10} mode lptn c[ode]p[age] prep[are]=((*charsets*)[*drive*]
[*path*\][*printer*.cpi)

{11} mode con c[ode]p[age] sel[ect]=*charset*

{12} mode lptn c[ode]p[age] sel[ect]=*charset*

{13} mode lptn c[ode]p[age] ref[resh]

{14} mode *device* c[ode]p[age] [/sta[tus]]

DOS Versions

{1}{5}	PC-DOS 1.1
{1}{3}{5}{7}	MS-DOS 2.1 to 3.2
{1}{3}{5}{7}	PC-DOS 2.0 to 3.2
{1}{3}{5}{7}{9}{10}{11}{12}{13}{14}	MS-DOS 3.3 to 3.31
{1}{3}{5}{7}{9}{10}{11}{12}{13}{14}	PC-DOS 3.3
{all}	MS-DOS 4.0
{all}	IBM DOS 4.0

Overview

The Mode command sets the operational mode for various devices. It is used to

- Select type styles on some types of printers
- Configure the serial port

- Direct output to a serial printer
- Choose a display mode for the screen

Many versions of DOS include additional functions and parameters for the Mode command, which is probably the least standardized of all the "standard" DOS commands. However, syntax forms 1, 3, 5, and 7 are present in almost all versions of DOS from 1.1 through 3.2. This discussion includes only the generally accepted forms.

After DOS 3.3, the Mode command is also used to control the use of national character sets on screens and printers. Specifically, it allows you to display these character sets on an EGA, VGA, or LCD display, and to print them on an IBM Proprinter II Model 4201 or an IBM Quietwriter III Model 5202.

When you use Mode to configure a serial port or a printer, and you ask it to check the status of the port as part of the command, part of its code remains in memory, leaving you less available RAM (see TSRs). You will see the message:

```
Resident portion of MODE loaded
```

Once it is loaded, issuing other Mode commands will not increase the amount of memory Mode occupies.

Procedure/ Examples

Mode is an external command; using it requires having the DOS program MODE.COM (or MODE.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:mode co80
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\mode lpt1=com1
```

In DOS 2, device names in the Mode command must end in a colon (see Device Names). In DOS 3, the colon is not needed.

Configuring the Serial Port

{1} mode com*n*:*bps*,*parity*,*databits*,*stopbits*[,p]

{2} mode com*n*:[baud=]*bps*,[parity=]*parity*,[data=]*databits*,
[stop=]*stopbits*

The serial ports are a means by which your computer can communicate with other devices (see Ports, Serial and Parallel). Among the common serial devices are plotters, mice, digitizers, modems, and some types of printers. The Mode command is used to structure the information passing through the serial port so that it makes sense to the receiving device.

Normally, information in the computer is grouped into *bytes*—eight-place binary numbers. Each place in a binary number is called a *bit*. Information sent to the serial port is transmitted as a series of bits, not bytes. These bits are transmitted in series, one bit at a time. Since every bit has a value of 0 or 1, the information passing through the port must be structured in some way; otherwise, it cannot be translated back into bytes. The Mode command formats the information into several types of bits.

Serial ports can receive as well as transmit information. Therefore, the structure you use to send information to the serial port is also used to decode the information that you receive from the port. The devices at both ends of the serial interface (one of which is your computer) must be prepared to use the same conventions for interpreting the bits.

The data format created by the Mode command consists of four elements:

- The start bit
- The number of data bits per byte
- The number of stop bits per byte
- The type of parity used

Transmitting a complete byte via the serial port requires a total of 10 bits. When a byte of data is transmitted, it is preceded by a single *start bit*, which is a binary 0. This is a "wake-up" signal to the receiving device. There is always one start bit. You do not have to set it.

The start bit is followed by either 7 or 8 *data bits*. If the information transmitted or received involves only text—that is, the first 127 characters in the ASCII code—you can convey all the necessary information in bytes of 7 data bits. (Seven bits are enough to hold binary representations of numbers through 127.) Otherwise, you must use 8 data bits.

If the data format uses 7 data bits, it is followed by a *parity bit*. Parity is a primitive means of checking transmitted data for errors.

If *even* parity is used, the parity bit receives a value of 1 when the sum of the 7 data bits is an odd number. When the sum of the data bits is an

even number, the parity bit receives a value of 0. If *odd* parity is used, the reverse occurs.

With 8 data bits, there is no parity bit. You must indicate this by setting parity to *none*.

DOS 4.0 includes the ability to set the parity bit to two other types of parity, neither of which has been in common use for some time: *mark* and *space*. With mark parity, the parity bit is always 0, regardless of the sum of the data bits. With space parity, the parity bit is always 7, regardless of the sum of the data bits.

The end of a transmitted byte is marked by either one or two *stop bits*. These are binary 1's, which represent the line signal's "resting" state. When a serial device detects stop bits, it waits until it receives the next start bit before it sends any further data to the computer. With 8 data bits, or with 7 data bits and a parity bit, 1 stop bit is used. With 7 data bits and no parity bit, 2 stop bits are used. The Mode command lets you set all these data format parameters as well as the rate of transmission, commonly—and incorrectly—called the *baud rate*. (The correct term is *bits per second*, or bps.)

Whether you need to configure the serial port depends on the device attached to it and its software. Most people use modems with telecommunications software that formats the data, bypassing the Mode command. Most serial mice come with driver programs that configure the serial port. Printers and plotters, however, must have the serial port configured for them. Also, if you plan to hook your computer to another computer via the serial port and use one as a dumb terminal, you will probably have to configure the serial ports on both to use the same data format and transmission speed (see Ctty).

The command to configure the serial port takes the form

```
mode comn:bps,parity,databits,stopbits
```

where: `comn` is the logical name of the port, *bps* is the transmission rate, *databits* is the number of data bits, and *stopbits* is the number of stop bits. With DOS 4.0, you may include the names of the items in the command, (as mnemonics to yourself), although they are unnecessary. Table 19 shows the acceptable values for these parameters.

If your software does not configure the serial port for the hardware attached to it, you must configure the port yourself. Consult the manual for the hardware attached to the port to find out the correct parameters. The command

```
mode com1:110
```

TABLE 19 **Acceptable Serial Port Parameters**

PORT NAME[1]	BAUD RATE	DATA BITS	STOP BITS	PARITY
COM1	110	7	2	o (Odd)
COM2	150	8	1	e (Even)
	300			n (None)
	600			m (Mark)[2]
	1200			s (Space)[2]
	2400			
	4800			
	9600			

1. *See Ports, Serial and Parallel, for information about logical ports with names higher than COM2.*
2. *Available in DOS 4.0 only.*

with no further parameters configures COM1 for even parity, 7 data bits, and 2 stop bits. If you specify any other transmission speed, the default is 1 stop bit. The other defaults are the same. You can refer to a transmission speed by its first two digits.

To configure the second serial port for 2400 bps, 7 data bits, odd parity, and 2 stop bits, use the command

```
mode com2:24,o,7,2
```

Redirecting Printer Output

{3} `mode com`*n*`:`*baud*`,`*parity*`,`*databits*`,`*stopbits*`[,p]`
`mode lpt`*n*`:=com`*n*`:`

{4} `mode com`*n*`:[baud=]`*bps*`,[parity=]`*parity*`,[data=]`*databits*`,`
`[stop=]`*stopbits*`[,[retry=]`*c*`]`
`mode lpt`*n*`:=com`*n*`:`

Configuring a serial printer requires an extra step. You must configure the port as described above, but you must also tell DOS to send printer output to a serial port instead of to a parallel port. (See Ports, Serial and Parallel, and Device Names). This requires two Mode commands. First, issue the command to configure the serial port. When configuring a serial

port for a printer or plotter, use the p parameter at the end of the command unless you are using DOS 4.0, in which case you use the b parameter. This tells DOS to continue trying to send output to the printer when the printer appears to be unavailable. Without the p parameter, printing will simply stop if a *timeout error* occurs.

With DOS 4.0, you have three choices of how to handle errors from the port, which you control with the retry= parameter. If you use the value of e, error messages will appear on the screen if the port is busy. This is sometimes helpful in network situations. The b value returns the actual status of the port, reporting it busy if it is. This setting is recommended for printer redirection. The r setting always reports that the port is ready to receive data, whether it is or not. This is actually what happens when you use the p parameter with earlier versions of DOS.

A typical pair of commands to configure the first serial port as the first printer port is

```
mode com1:12,n,8,1,p
mode lpt1:=com1:
```

In DOS 3.0 and later, you can cancel the second command by entering the command

```
mode lpt1
```

This causes the output destined for LPT1 to go to the first serial port instead of elsewhere. In DOS 2, the only way to cancel the command is to reboot the computer.

If you have a serial printer, you should put the commands that configure it in your AUTOEXEC.BAT file (see AUTOEXEC.BAT File).

Configuring the Printer

{5} mode lpt*n*:[*chars*][,[*lines*][,p]

{6} mode lpt*n*:[cols=*chars*][,[lines=*lines*][,retry=*c*]

If you have an IBM, Epson, or Epson-compatible printer, you can configure some aspects of the printer output with the Mode command. The general form is

```
mode lptn:chars,lines,p
```

where *n* is the number of the serial port, *chars* is the number of characters per line (80 or 132), and *lines* is the number of lines per inch (6 or 8). The

defaults are 80 characters per inch and 6 lines per inch. The p parameter tells DOS to keep trying to send output to the printer when the printer appears to be unavailable. If you omit the p, printing will simply stop when a *timeout error* occurs.

With DOS 4.0, you can include the names of the items in the command, as mnemonics to yourself, although they are unnecessary. Here, as in other commands which formerly used the p parameter, you have a choice of three ways for DOS to deal with a busy port, represented by the e, b, and r values for retry=.

Configuring the Screen

{7} mode *display*
{8} mode con [*display*] [[cols=]*chars*][,[[lines]=*lines*]

Your computer may be able to use several different *display modes* depending on your hardware. Table 20 shows the mode parameters for various display modes and the types of adapters with which you can use them. If your screen doesn't display 80 columns of text properly, you can use the mode command to adjust it. Enter the command

```
mode display,t
```

where *display* is one of the modes from Table 20. The t parameter displays a test pattern of 40 or 80 digits (depending on the mode you choose) and one of the following messages:

```
Do you see the leftmost 0 (Y/N)?
Do you see the rightmost 1 (Y/N)?
```

TABLE 20 **Display Mode Parameters**

MODE NAME	DISPLAY	ADAPTER TYPE
mono	monochrome 80-column text	Monochrome
co40	40-column color text	CGA, EGA, or VGA
co80	80-column color text	CGA, EGA, or VGA
bw40	40-column black & white text	CGA, EGA, or VGA
bw80	80-column black & white text	CGA, EGA, or VGA

If you type Y, the series of digits will step one place to the right or left. If you have performed this test and know which direction to shift the pattern, you can enter it as a parameter, such as

```
mode bw80,r
```

This sets the screen to 80 columns, with color disabled, and the display shifted one column to the right.

If you use DOS 4.0 with an EGA or VGA display adapter, and you have loaded ANSI.SYS with the /l switch, you can use the Mode command to set the number of lines that appear on your screen. With the EGA you can have the standard 25 or 43; with the VGA, 25 or 50. You have a choice of several forms of the Mode command to accomplish this:

```
mode display lines=lines
mode lines=lines
mode lines
mode lines=lines,rows=chars
mode lines,chars
```

Since the acceptable values for *lines* are 25, 43, and 50, and those for *chars* are 40 and 80, you can safely enter either without confusion. Also, because most forms of the *display* parameter include a number of characters per line, you needn't specify this parameter a second time. (In fact, this command often doesn't work.)

If the number of lines you have established interferes with your application programs, rewrite your CONFIG.SYS file to remove the /l switch from the command that loads ANSI.SYS. See CONFIG.SYS File and ANSI.SYS for details.

Mode commands that select the display format do not place any of Mode's code in memory.

Alternative Character Sets—General Principles DOS 3.3 or later

allows you to use any of several national character sets. Table 21 shows the character sets available and the character set *code page numbers* used to activate them. With earlier versions of DOS, you could not select the character set you wanted to use because it was built into ROM for each country. Under DOS 3.3 and later, all languages that do not use the default character set (437) are expected to use the international character set (850). However, to view or edit files created using DOS 3.0 to 3.2 in one of

TABLE 21 **Code Numbers for Alternative Character Sets**

CHARACTER SET	CODE
United States	437
International	850
Portugal	860
French Canada	863
Norway, Denmark	865

the other four languages represented, you must be able to switch to the old character set. Accomplishing this takes three to five steps, depending on your hardware.

First, include several commands in your CONFIG.SYS file. Begin by loading ANSI.SYS (see ANSI.SYS). If you have an EGA, VGA, or LCD display, load DISPLAY.SYS (see DISPLAY.SYS). If you have one of the two types of printers that can print the alternative character sets, load PRINTER.SYS (see PRINTER.SYS). (See also ChCP for further details.)

If you have a monographics or color graphics adapter and use any programs that display text on a graphics screen, you must also issue a GrafTabl command (see GrafTabl).

Next, execute NLSFunc (see NLSFunc). No one seems to know what this mysterious command does, but you can't complete the next step if you don't load it.

Finally, issue the appropriate Mode CodePage Prepare command, described next. Once you have done so, you can use the other Mode commands, whose descriptions follow, to manage the character sets. All the commands can use either full or abbreviated syntax. The full syntax is easier to remember, but the abbreviated syntax is easier to type.

{**9**} `mode con c[ode]p[age] prep[are]=((`*charsets*`)[`*drive*`][`*path*`\][`*display*`.cpi)`

{**11**} `mode con c[ode]p[age] sel[ect]=`*charset*

If you have taken the necessary preliminary steps, you can employ the alternative character sets using commands of the form

```
mode con codepage prepare=((charsets)display.cpi)
```

or

```
mode con cp prep=((charsets)display.cpi)
```

where *charsets* is one or more of the code numbers from Table 21, and *DISPLAY*.CPI is a code page information file. There are two such files for the screen: EGA.CPI for the EGA and VGA, and LCD.CPI for LCD displays. If these files, included in DOS since release 3.3, are not on the current directory, preface the file name with the drive and/or path name.

The number of character sets you can load depends on the number you prepared in the `device=display.sys` command in the CONFIG.SYS file. Thus, if you prepared for two character sets and have an EGA display, you can use the command

```
mode con cp prep=((850,865)c:\dos\ega.cpi)
```

to prepare to switch between the international and Danish character sets. If you are successful, you will see the message

```
MODE codepage prepare function completed
```

You can change which character sets are prepared by issuing new `mode...cp prep` commands, replacing the code numbers for those character sets you want to change. Thus, to replace the Danish character set with the Portuguese, but keep the international set, enter the command

```
mode con cp prep=((,860)c:\dos\ega.cpi)
```

Preparing for Alternative Character Sets on the Printer

{**10**} `mode lptn c[ode]p[age] prep[are]=((`*charsets*`)[`*drive*`][`*path*`\][`*printer*`.cpi))`

{**12**} `mode lptn c[ode]p[age] sel[ect]=`*charset*

{**13**} `mode lptn c[ode]p[age] ref[resh]`

If you have an IBM Proprinter II Model 4201 or an IBM Quietwriter III Model 5202, you can print files using the alternative character sets. The number of character sets you can use depends on your hardware and on the `device=printer.sys` command in your CONFIG.SYS file. The principles are the same as for preparing the screen. If you have taken the necessary preliminary steps, you can print the alternative character sets using commands of the form

```
mode lptn codepage prepare=((charsets)display.cpi)
```

or

```
mode lptn cp prep=((charsets)display.cpi)
```

where *n* is the number of the parallel port to which the printer is attached, *charsets* is one or more of the code numbers from Table 21, and *PRINTER*.CPI is a *code page information file*. There are two such files for printers included in DOS releases since 3.3: 4201.CPI for the Proprinter II and Proprinter XL printers, and 5202.CPI for the Quietwriter III. DOS 4.0 added 4208.CPI for the Proprinter X24 and XL24. If the file you need is not on the current directory, preface the file name with the drive and/or path name.

If you prepare for two character sets in your CONFIG.SYS file and have a Proprinter attached to your second parallel port, you can use the command

```
mode lpt2 cp prep=((437,863)c:\dos\4201.cpi)
```

to prepare to switch between the English Canadian (U.S.) and French Canadian character sets. If you are successful, you will see the message

```
MODE codepage prepare function completed
```

You can change which character sets are prepared by issuing new `mode...cp prep` commands, replacing the code numbers for those character sets you want to change. Thus, to replace the English Canadian character set with the international, but keep the French Canadian set, enter the command:

```
mode lpt2 cp prep=(850,)c:\dos\4201.cpi)
```

If your printer is attached to the first parallel port, you can use PRN in place of LPT1.

Using the Alternative Character Sets

{**11**} `mode con c[ode]p[age] sel[ect]=`*charset*
{**12**} `mode lpt`*n* `c[ode]p[age] sel[ect]=`*charset*
{**13**} `mode lpt`*n* `c[ode]p[age] ref[resh]`

Once you have prepared the character sets, using the NLSFunc and `mode...cp prep` commands to place them in memory, you can switch among them using the select modifier. If you have issued the appropriate `cp prep` command, you can make the French Canadian character set current on your printer by entering the command

```
mode lpt2 codepage select=863
```

or

```
mode lpt2 cp sel=863
```

The Mode command switches among alternative character sets in a printer by turning them on or off in the printer's character ROM. If you turn the printer off, it will forget which character set it was using and

return to its default state when you turn it on. Use the Refresh modifier to retrieve the character set you have prepared when it fades away:

```
mode lptn codepage refresh
```

or

```
mode lptn cp ref
```

If you have prepared the international character set for your screen, you can make it current by entering the command

```
mode con codepage select=850
```

or

```
mode con cp sel=850
```

Checking the Status of the Character Sets

{**14**} mode *device* c[ode]p[age] [/sta[tus]]

To find out which character sets are prepared and which one is selected, enter the Mode command, followed by the logical name of the device about which you want to inquire. For example, to check on the status of your screen fonts, enter

```
mode con cp
```

or

```
mode con cp /sta
```

or

```
mode con codepage /status
```

or any combination of these commands. You will see a display similar to Figure 79.

To check on a printer, replace con with the logical name of the printer. If you have made an error or failed to take all of the preliminary steps, you will see the message

```
Device error during Status
```

```
Active codepage for device CON is 850
hardware codepages:
  Codepage 865
prepared codepages:
  Codepage 437
  Codepage 850
  Codepage 865
MODE Status Codepage function completed
```

Figure 79: Codepage status display

Tips

If you have two display adapters and monitors in your system, use a Mode command in your AUTOEXEC.BAT file to select the monitor you want to be active when you start up your system.

You can switch between a monochrome and a color monitor by issuing the commands `mode mono` and `mode co80`, respectively. These commands will not clear the screen from which you are switching. Therefore, you can use this technique to view a portion of one file while you edit another, or to view data in a spreadsheet while you fine-tune a graph based on the data.

For full information about serial and parallel ports, see Device Names, and Ports, Serial and Parallel. For further details about using alternative character sets in DOS 3.3 and later, see ChCP, DISPLAY.SYS, PRINTER.SYS, NLSFunc, and GrafTabl. For information about using these alternative character sets with keyboard layouts that can produce the characters, see Keyb and KEYBOARD.SYS.

More

Syntax

{1} `type` *`textfile.txt`*`|more`
{2} `more<`*`textfile.txt`*

DOS Versions

MS-DOS 2.11 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

More is a filter program that stops output to screen from scrolling after the screen fills, allowing you to view text files one file at a time (see Filters). When the screen fills, the message

```
-- More --
```

appears in the lower left corner, and the display does not change until you press a key.

Procedure/ Examples

More is an external command; using it requires having the DOS program MORE.COM on a disk in your computer. If you use DOS 2, it must be

either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:more<a:report.txt
```

If you use DOS 3.0 or later, and the program is neither in the current directory or on the search path, preface the command with a drive name and/or path name; for example,

```
type a:report.txt¦c:\dos\more
```

{1} `type` *`textfile.txt`*`¦more`
{2} `more<`*`textfile.txt`*

To view a file a screen at a time, you can either pipe the output of the Type command to the More filter by entering a command such as

```
type carlsen.ltr¦more
```

or redirect the file to the More filter as input by entering a command such as

```
more<carlsen.ltr
```

For more information on the structure of these commands, see Redirection and Pipes.

The effect of either command is the same when you look at the screen, but the difference between them is more than theoretical. When you send the output of the Type command to the More filter, you create an intermediate file, or *pipe*, on the default disk, which takes a little more time than using redirection without creating an intermediate file. It also means that the command will fail if the default disk is write protected since there will be no place for DOS to create the intermediate file.

Warnings

The More command is subject to the same limitations as the Type command in the types of files and characters it can display (see Type).

Tips

You can use More with any command that displays text on the screen. For example,

```
tree /f¦more
```

gives you a listing of all the files on the default disk, directory by directory, displayed a screen at a time, as will:

```
chkdsk /v¦more
```

You can create a command that displays files a screen at a time by creating the batch file VIEW.BAT, containing the single line:

```
more <%1
```

(See Batch Files for an explanation of the replaceable parameter %1.)

See the programs in Call; Hard Disks, Backing Up and Restoring; or Redirection and Pipes regarding ideas about using the More command to append a carriage return to a redirected file.

For more general information, see Filters, and Redirection and Pipes. See also Type for information about displaying files.

NLSFunc

Syntax

{**1**} nlsfunc
{**2**} nlsfunc [*drive*][*path*\]country.sys
{**3**} install=nlsfunc.exe

DOS Versions

{**1**}{**2**} MS-DOS 3.3 to 3.31
{**1**}{**2**} PC-DOS 3.3
{**all**} MS-DOS 4.0
{**all**} IBM DOS 4.0

Overview

NLSFunc stands for national language support function. You must enter this command before you can use the ChCP command to switch to alternative national character sets. If you don't, the ChCP command won't work. NLSFunc is a TSR that occupies about 2½K of your RAM (see TSRs).

Procedure

NLSFunc is an external command; using it requires having the DOS program NLSFUNC.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\nlsfunc
```

{**1**} nlsfunc
{**3**} install=nlsfunc.exe

To use the ChCP command to switch between national-language fonts, you must first execute NLSFunc. Do so by entering the command:

```
nlsfunc
```

If you use DOS 4.0, you can load NLSFunc via your CONFIG.SYS file, by using the command

```
install=nlsfunc.exe
```

Preface the file name with a drive and/or path name if the file is not on the root directory of your boot disk. See Install= and CONFIG.SYS File for details.

{2} `nlsfunc [drive][path\]country.sys`

If you forgot to include a Country= command in your CONFIG.SYS file and want to use alternative language fonts and keyboards, you can tell DOS where the country information file is by including it in the NLSFunc command. Some of the DOS documentation implies that you can choose national defaults while entering the NLSfunc command with this syntax, although you can't.

If you invoke NLSFunc more than once, you will see the message:

```
NLSFUNC already installed
```

Examples

Assuming you have loaded the CONFIG.SYS file shown in the DISPLAY.SYS entry and you have an enhanced graphics adapter, you can complete configuring your computer for Belgian national defaults by including the code that follows in your AUTOEXEC.BAT file.

AUTOEXEC.BAT Code to Nationalize the Computer for Belgium

COMMANDS	COMMENTS
`nlsfunc`	Invokes national language support
`keyb be`	Loads the Belgian keyboard layout into memory
`mode con cp prep=((850)ega.cpi)`	Prepares the screen to use Code Page 850 (the international character set)
`chcp 850`	Switches the screen to the international character set

Once these lines are executed, you can switch back to the normal keyboard by pressing Ctrl-Alt-F1, then to the Belgian keyboard by pressing Ctrl-Alt-F2. You can also switch back to the standard font by issuing the command:

```
chcp 437
```

See Table 21 in the Mode entry for a list of code page (character set) numbers.

Tips

For complete information on teaching your computer to speak another language, see also ChCP, Country=, COUNTRY.SYS, DISPLAY.SYS, GrafTabl, Keyb, Mode, and PRINTER.SYS.

NUL

Overview

NUL is the name given to the null device, also sometimes known as the "bit bucket." Any output directed to NUL will be discarded. See Device Names, and Redirection and Pipes for further information.

Path

Syntax

{**1**} `path [drive]\path;[drive]\path; . . .`
{**2**} `path=[drive]\path;[drive]\path; . . .`
{**3**} `set path=[drive]\path;[drive]\path; . . .`
{**4**} `path`

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

A Path statement extends the scope of DOS's search for executable files. Normally, when you issue a command without specifying the drive or directory where a program is located, DOS looks in COMMAND.COM for internal commands, then looks in the current directory. If the command is not in either of these places, it displays the message `Bad command or file name`. (For elaboration on this point, see Command Processing Order.)

In DOS 2, if you precede the command with a drive name, DOS looks only in the default directory of the specified drive before displaying the same message. DOS 3 lets you specify a drive and path, in which case DOS will look only in the specified directory.

When you enter a Path command, however, you tell DOS to look in several other directories before giving up and displaying the message. This is true in all versions since 2.0. In DOS 2, you can override this search by specifying a drive, in which case it will look in the current directory of the drive. In DOS 3.0 and later you can also override the search by specifying a path name, including a drive name if necessary (see Path Names).

Thus, the Path command lets you execute programs that are not in the current directory. With DOS 2, you have to make the directory containing the program current before you can execute the program when you have not entered a Path command. With DOS 3.0 or later, you must enter either the path from your current location to the program, or the complete path name when you have not entered a Path command.

Procedure

Setting a Search Path

{**1**} `path [drive]\pathname;[drive]\pathname; . . .`
{**2**} `path=[drive]\pathname;[drive]\pathname; . . .`
{**3**} `set path=[drive]\pathname;[drive]\pathname; . . .`

To set a search path, enter a Path command followed by a list of the names of the paths you want DOS to search, separated by semicolons. A semicolon at the end of the list is optional. Do not leave any spaces between the path names. All these syntax forms work the same way. Except under unusual circumstances, you should place your Path command close to the beginning of your AUTOEXEC.BAT file, although you can enter a Path command at a DOS prompt (see AUTOEXEC.BAT File).

You may precede any path name with a drive name. If you do not precede a path name with a drive name, DOS looks for the specified directory only on the default disk. This can be annoying if you use a hard disk and frequently make other drives current. Under these circumstances, include the drive names with your hard disk directories.

When you enter a new Path command, the new search path replaces the old one.

Examining the Search Path

{**4**} `path`

To see a list of directories currently on the search path, type `path` with no parameters. You will see a list displayed in syntax form 2. You will also see this list, along with some other information, if you type `set`.

Length of the Search Path The maximum length of a DOS command depends on the version of DOS you use. Normally, a DOS command cannot exceed 128 characters (although some versions I have used permit 132 or 137 characters). This limits the number of directories you can place on the search path. If by some chance you manage to enter a path command that contains more than the number of characters allowed by your DOS—perhaps by means of a batch file—you will see a `Bad command or file name` message.

The more directories you place on the search path, the longer it takes DOS to complete a search before giving up and displaying a `Bad command or file name` message. You can decrease this time by using the Buffers= command to increase the number of DOS memory buffers (see Buffers=).

PATH is an environment variable (see Environment Space, and Set). This means that if your search path gets too long, you may see the message `Out of environment space`. This is especially likely to happen if any of the following is true:

- You have a complex prompt
- You load DOS-resident programs before entering your Path command (see TSRs)
- You use DOS 3.3 or later and keep your Append list in the environment (see Append)

To overcome this problem, either shorten your path as suggested under Tips, or expand your environment space with the Shell= or Command commands (see Shell= and Command).

What to Put on the Path What you put on your search path depends largely on two factors: the number and types of drives in your system, and your style of working.

If you do not have a hard disk, your need for a search path is limited. However, if you also use a RAM disk (see VDISK.SYS, AUTOEXEC.BAT File, and RAMDRIVE.SYS), you can simplify your work a bit by including the statement

```
path=c:\;a:\
```

in your AUTOEXEC.BAT file. This assumes that you use drive B for data diskettes.

If you have a hard disk, you should always use a search path. At the very least, you should have the following Path statement in your AUTOEXEC.BAT file:

```
path c:\dos;c:\batch;c:\utility
```

This will enable you to execute DOS commands and utility programs from any directory on any drive. If you construct the \BATCH directory as described in the Directories entry, you will be able to execute most of your programs from any point in your system, even if they are not on the search path (see Directories).

If you keep your data files in their own directories rather than in directories with the software that created them, you might also include your software directories in the Path statement. Whether you do so depends largely on the characteristics of the software. Many programs can be executed from anywhere in the system if their directory is on the search path (among them Framework II and III, Microsoft Word, dBASE III PLUS, and WordStar 4.0). You can benefit greatly by placing such programs on the path. If you use some of these programs, you might extend your path to include them:

```
path c:\dos;c:\batch;c:\utility;c:\word;c:\dbase;c:\framewk
```

Some programs that use overlays cannot be executed in this manner unless you also append their directories (see Append). Perhaps most notable for this failing is WordStar 3.3. Other programs cannot be invoked except from their home directory.

WordPerfect is perhaps the only major program that you can call from anywhere in a system without placing it on a search path. If you enter a command of the form

```
c:\path\wp
```

it will execute reliably, regardless of what drive or directory is current.

Some utility packages include so many files that you would lose track of them if you put them in a \UTILITY directory. You should give these packages their own directories, and put them on the path as well.

As a general rule, place on the search path any directory containing software you want to call from anywhere in your system. If that software cannot load itself with its directory on the path, it may need help from the Append command. If you use DOS 3.0 or earlier, see Tips for alternatives to the Append command.

Warnings

No matter what you put on the search path, it will not affect files loaded by the CONFIG.SYS file (see CONFIG.SYS File). This is because Path is a DOS command and is not executed until all the files in CONFIG.SYS have already been loaded. Therefore, you must include a complete path name for any program loaded by CONFIG.SYS that is not in the root directory of the boot disk.

You may be tempted to include the root directory of all your drives in the search path, including your diskette drives. If you do most of your work on a hard disk, this can cause you more trouble than it's worth. Every time DOS searches the path, it will attempt to read the directories of the disks it expects to find in the diskette drives. If those drives are empty, you will be faced with the message

```
Not ready error reading drive A:
Abort, Retry, Ignore?
```

or, in DOS 3.3 and later, the even more chilling

```
Abort, Retry, Fail?
```

every time DOS searches the path.

Tips

Path is an environment variable (see Environment Space, and Set). Therefore, you can change its value in a batch file, just like any other environment variable. For example, to add directories to the path while you are working, use the batch file MOREPATH.BAT, containing the line

```
set path=%path%;%1
```

To add a new path to the list, for example, a directory called C:\NEWPROG, enter the command

```
morepath c:\newprog
```

If your path is the one shown above, it becomes

```
path c:\dos;c:\batch;c:\utility;c:\word;c:\dbase;c:\framewk;c:\newprog
```

If your path gets too long, you can either use short directory names or use the Subst command (see Subst). If instead of \BATCH, \UTILITY, \WORD, \DBASE, and \FRAMEWK you used \BAT, \UT, \WP, \DB, and \FW, your path would read:

```
path c:\dos;\c:\bat;c:\ut;c:\wp;c:\db;c:\fw
```

This path is thirteen characters shorter than the original. Of course, you must then remember what all those two-letter abbreviations stand for, and you may eventually run out of useful ones.

The Subst command lets you substitute a drive name for a path name. If you want to include on your path several directories that are children of a single directory, use the Subst command to give the parent directory a drive name before you enter the Path command. This will shorten the path considerably.

The Path command affects only executable files. It does not affect files named as parameters to commands, nor does it affect overlay files, driver files, or other ancillary files used by executable programs. For such files, you must use the Append command (see Append). If you use DOS 3.0 or earlier, you can use commercial *path-extension* software to gain the same benefits.

For related information, see Append, Directories, and Path Names.

Path Names

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

When you use tree-structured directories, DOS locates files by their *path names*. A path name is a list of the directories between the root, or the current directory, and the directory containing the file. (For more information about tree-structured directories, see Directories.) Each directory name in a path name is separated from those above and below it by a backslash. When you refer to a file with a path name, the file name is also separated from its directory name by a backslash.

In the following discussion, assume that the hard disk is drive C. The directory structure is the sample shown in Figure 28 in the Directories entry. You may use Figure 28 for reference.

Procedure/ Examples

Absolute and Relative Path Names A path name may be absolute or relative. An *absolute* path name is a list of all the directories from the root directory to the target directory. A *relative* path name is a list of all directories from the current directory to the target directory. The root directory is always designated by a backslash alone (\).

The basic rule in using path names is that you must give DOS enough information to find the directory you want to refer to. If you are some distance from the desired directory in the tree structure—say, if you are in \INVOICES\MIDWEST\QTR1, and you want to refer to \REPORTS\EAST\QTR3—it is easiest to use the absolute path name in your command:

```
\reports\east\qtr3
```

If the directory you want to refer to is a child of your current directory—for example, if your current directory is \REPORTS\EAST and you want to refer to \REPORTS\EAST\QTR3—you could use the path name

```
qtr3
```

Note that you do not use a leading backslash (a leading backslash always refers to the root directory). If you did use a leading backslash, DOS would expect the QTR3 directory to be directly subordinate to the root.

You can refer to the parent of the current directory in a command by using the double period. Thus, if your current directory (see Figure 28) is \INVOICES, the command

```
dir ..
```

will display the root directory. You must separate the double period from the command with a space in DOS 2, but need not in later releases.

You can use the double period in longer relative path names. Thus, you can use the .. directory name to move through levels if you have subdirectories nested several levels deep. If you want to refer to \REPORTS\MIDWEST\QTR3, while logged onto \REPORTS\MIDWEST\QTR1 you could do so by using the path name:

```
..\qtr3
```

This translates as, "Up one level from the current directory, and then down one level to QTR3." To address a directory even farther away—say, \INVOICES\SOUTH\QTR1—you could use the path name:

```
..\..\..\invoices\south\qtr1
```

The three double periods refer to MIDWEST, REPORTS, and the root, respectively. Of course, it would be much easier to just use the absolute path name:

```
\invoices\south\qtr1
```

Treating the Current Directory as a Path There is also a shorthand notation for addressing the current directory. You can refer to the current directory in a command by representing it with a period. Thus, if drive C is

current, either of the following commands will copy all the files from the current directory of drive C to the current directory of a disk in drive A:

```
xcopy . a:
xcopy c: a:
```

In DOS 2, you must include a space between the command and the period. In later releases, you don't need the space.

Using Path Names in Commands There are three special commands for creating, selecting, and deleting directories. The MD (also called MkDir) command creates new directories; the CD (also called ChDir) command selects a directory as the current directory; the RD (also called RmDir) command deletes a directory. You can use relative or absolute path names with all these commands, including the double-period syntax.

With these three commands, you need not enter a space between the command and the path name if the path name begins with the root directory. However, with other commands, you must always separate the command from the path name with a space.

At the DOS prompt, you can use path names as parameters to commands. Thus, `dir \invoices\south` displays the directory of the \INVOICES\SOUTH directory, regardless of which directory is current.

The path name can include a file name. For example, to read a file called SALES.DOC on the directory \REPORTS\MIDWEST\QTR2, you could use the command:

```
type \reports\midwest\qtr2\sales.doc
```

Alternatively, you could log onto the directory and simply enter the command:

```
type sales.doc
```

In DOS 3.0 and later, you can also execute commands on noncurrent directories by including a path name (absolute or relative) before the command. For example, if the root directory is current, you can format a diskette by using the command

```
c:\dos\format a:
```

This technique does not work in DOS 2. The only way to execute a command on a noncurrent directory in DOS 2 is to include the directory on a search path (see Path).

Warnings

The single-period reference to the current directory does not work in commands that require a file name. The command

```
copy . a:
```

is acceptable because you can ask the Copy command to copy everything in a directory by giving it the name of the directory. However, the command

```
replace . a:
```

results in this message:

```
No files found 'C:\'
```

Tips

Some software packages cannot be invoked by specifying the path to their directory. If a software package includes overlay files, special screen, keyboard, or printer drivers, or other nonexecutable files, it may not be able to find those files unless its directory is current or you place it in a Path and/or Append statement. See Path and Append for further information.

Pause

Syntax

{**1**} pause
{**2**} pause *message*

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

Pause is a batch command that allows you to interrupt execution temporarily. When DOS encounters a Pause command in a batch file, it stops executing the file, displays the message

```
Strike a key when ready . . .
```

and waits for a keystroke. It can also display an additional message that you create. It is useful when you have to do something to a peripheral device before proceeding or when you want an opportunity to halt the program.

Procedure/ Examples

The action of the Pause command depends on the current state of the Echo command (see Echo). If Echo is on, you can include a message in the Pause command; if Echo is off, you cannot. In either case, the message

```
Strike a key when ready . . .
```

will appear.

To include a message, make sure Echo is on, and include the message as part of the Pause statement:

```
pause Insert data disk in drive B
```

This statement results in the display

```
pause Insert data disk in drive B
Strike a key when ready . . .
```

If you want to display a message when Echo is off, use the Echo command along with the Pause command:

```
echo Insert data disk in drive B
pause
```

The same message appears, but the Pause command itself does not appear on the screen. You might find this form somewhat more elegant than including the message in the Pause command.

You can use the Pause command when you want the user to turn on a peripheral device before a program accesses it:

```
echo Turn printer on and align paper.
pause
dir >prn
```

To give the user a chance to end the program, you can use either of the following forms. If Echo is on, use

```
pause Press Ctrl-C to end the program. Otherwise,
```

This will display the text

```
pause Press Ctrl-C to end the program. Otherwise,
Strike a key when ready . . .
```

If Echo is off, use

```
echo Press Ctrl-C to stop. To continue,
pause
```

Tips When you need to tell the user to do something, Ctrl-G can be added to the message to sound a beep:

```
echo ^GTurn printer on and align paper.
```

Be careful not to overuse beeps, however; they can quickly become annoying.

Ports, Serial and Parallel

Overview *Ports* are the connectors through which your computer communicates with other peripheral devices. DOS computers use three types of ports: *serial* ports, *parallel* ports, and *game* ports. Depending on the brand of computer, some of these ports may be built in. Many types of expansion boards include ports of various types.

Serial Ports Serial ports, also known as *communications ports*, transmit information a single bit at a time. Before they can transmit information, therefore, each 8-bit byte must be broken down into a series of bits. Since one bit looks like any other to the computer (it is either a 1 or a 0), bits must be *formatted* into meaningful groups. To configure the data format, use the Mode command (see Mode).

Serial ports can both transmit and receive information. They are commonly used for modems, plotters, some types of mice (and other digitizing devices), and some types of printers.

Most MS-DOS versions before 3.3 can address a maximum of two serial ports, although it is sometimes possible to install as many as four ports. The serial ports have the logical names COM1 or COM2. On computers in the PC, PC XT, and PC AT family, you can usually configure the ports and the boards containing them to use either logical name. You should configure the device attached to the port to use the same name as the port.

Internal modems do not require a separate physical port, but they do require a logical port name. Most internal modems can be configured as either COM1 or COM2. Some can also be configured as COM3 or COM4. However, many telecommunications software packages cannot address COM3 at present.

PS/2 computers can have up to eight serial ports, all of which can be addressed by DOS 3.3. These computers automatically assign each port a logical name from COM1 to COM8.

The logical device names AUX and USER are sometimes used for serial ports (see Device Names).

Parallel Ports Unlike serial ports, parallel ports transmit all 8 bits of a byte at once, along eight parallel electrical paths (hence, the name). MS-DOS can address three parallel ports, using the names LPT1, LPT2, and LPT3. Because printers are usually the devices attached to a parallel port, the name of this port stands for line printer. Thus, you can often address LPT1 as PRN or PRT, depending on your version of DOS.

You must inform DOS if you have a serial printer and redirect printer output to a serial port using the Mode command (see Mode).

Most MS-DOS computers can only output to parallel ports. However, PS/2 computers can also receive input through them. See Copy35 and Rcv35 for further information.

Game Ports Usually, the only type of device attached to a game port is a joystick. However, some digitizing tablets also use a game port. As a rule, the game port is configured by the software that uses it.

Tips

Although most DOS computers recognize only two serial ports, COM1 and COM2, it may be possible to install a third if you are able to assign one of the ports to a different IRQ (interrupt request) at the BIOS level. COM1 normally uses IRQ 4 and COM2 IRQ 3. If you can determine which interrupt your third serial port is using, you may be able to reassign it to IRQ 2 (on a PC, PC XT, or compatible) or IRQ 9 (on a PC AT or compatible). If so, you can install it as COM3. Whether this is possible depends on the following things:

- The board on which your serial port is installed. Some boards have jumpers or switches that allow you to select the interrupt, but others do not.

- The device attached to the port. Some devices can be set to use COM3 or COM4, but others cannot.
- The software that addresses the device attached to the port. Many programs do not recognize COM3 or COM4 as a legitimate port address.

Further details are beyond the scope of this book.

Print

Syntax

{1} print *file1.ext file2.ext . . . file10.ext*
{2} print *file3.ext* /c *file4.ext file10.ext*
{3} print *file11.ext* /p *file12.ext*
{4} print /t
{5} print
{6} print *file1.ext file2.ext . . . filen.ext* [/d:*dev*] [/m:*m*] [/b:*s*] [/u:*b*] [/s:*t*] [/q:*f*]

DOS Versions

{1}{2}{3}{4}{5}	MS-DOS 2.1 to 2.11
{1}{2}{3}{4}{5}	PC-DOS 2.0 to 2.1
{all}	MS-DOS 3.0 to 4.0
{all}	PC-DOS 3.0 to 3.3
{all}	IBM DOS 4.0

Overview

The Print command sets up a *print queue*—a list of files to print from disk. Print is a TSR that stays in memory until you reboot your computer (see TSRs). Unless you make a serious error, the files will continue to be printed while you perform other computer operations. This is the only form of multitasking provided by DOS.

It is difficult to give an overview of the Print command because it varies greatly from one version of DOS to another. The fifth syntax form includes six optional switches, not all of which are available in all manufacturers' versions of DOS. A perusal of four different DOS manuals shows four different sets of switches and four different sets of dos, don'ts, and cautions.

A few things seem to be true of all versions, however. First, the Print command prints files from disk. You give it a list of files to print, and it prints them in the order named. Once you activate Print, you cannot send anything else to the printer. If you try, you will get a `Printer not ready` message. You can, however, remove some files from the list, add others, or cancel the entire operation.

Second, you should never remove the disk containing the files to be printed. You risk serious loss of data if you do so.

Third, you cannot impose any kind of formatting on the material to be printed. Files not already formatted for printing will be printed "as is." They won't even skip the perforations on continuous-form paper.

Fourth, Print sends a form feed to the printer at the end of each file.

The versions of Print released with DOS 2 are more uniform and more limited.

Procedure/ Examples

Print is an external command; using it requires having the DOS program PRINT.COM (or PRINT.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
b:print textfile.txt textfile.doc
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\print \reports\report1.doc \db\report2.prt a:temp.txt
```

Unless otherwise specified, what follows applies to all releases of Print.

Setting Up a Queue

{**1**} `print` *`file1.ext file2.ext . . . file10.ext`*

In DOS 2, you enter the Print command followed by a list of the names of the files you want to print; you can include up to ten files. (You can increase this number in DOS 3.0 and later, as will be explained.)

You can use wildcard characters in the file names, but only ten files will be placed in the queue, regardless of how many names match (see Wildcard Characters). The maximum number of characters in the queue is

sixty-three. Thus, if wildcards in the file names will increase the number of characters to more than sixty-three, those files whose names appear after the sixty-third character will not be included.

You are then prompted with the line:

```
Name of list device [PRN]
```

If your printer can be addressed by the name PRN, simply press Return (see Device Names). If, for example, your printer is attached to LPT2 or COM1, you should enter the name of the port to which the printer is attached before you press Return.

In DOS 2, Print will print only files that are in the current directory when you issue the command. You can change to another directory once you have entered the Print command.

In DOS 3.0 and later, you can use the same syntax to set up the queue. However, you can also specify full path names to any of the files to be printed. On the other hand, the sixty-three-character limitation still applies. This may reduce the number of files you can print, unless you use some of the options described in the following sections.

Removing Files from the Queue

{2} `print` *`file3.ext`* `/c` *`file4.ext file10.ext`*

To remove a file from the queue, enter the Print command, followed by the name of the file to remove, followed by the /c (cancel) switch. To remove additional files from the queue, place their names *after* the /c switch. If you place ten files in the queue with the original Print command, with names *`file1.ext`* through *`file10.ext`*, the above command would remove the third, fourth, and tenth files from the queue. The others would still be printed.

Adding Files to the Queue

{3} `print` *`file11.ext`* `/p` *`file12.ext`*

The /p (print) switch adds files to the queue. Like the /c switch, you place it after the first of the files to add to the queue. You can add more files by placing their names after the switch. The above command will add two new files to the queue if there is room for them. If not, your screen displays the message `Print queue is full`.

You can add and remove files from the queue with a single command, but be careful where you put the switches. The two previous commands could be combined as follows:

```
print file3.ext /c file4.ext file10.ext file11.ext /p file12.ext
```

The Print command is smart enough to read backward. Thus, although the file *file11.ext* appears after the /c switch, it is also the one immediately *before* the /p switch, and it will be printed.

Canceling the Queue

```
{4} print /t
```

To stop the printing of files in the queue immediately, use the /t (terminate) switch. Printing immediately stops, even in the middle of the file. The message

```
All files canceled by operator.
```

is then printed.

Examining the Queue

```
{5} print
```

To see a list of the files waiting to be printed, enter the Print command with no parameters.

Controlling the Queue

```
{6} print file1.ext file2.ext . . . filen.ext [/d:dev][/m:m]
      [/b:s][/u:b][/s:t][/q:f]
```

Versions of Print released with DOS 3.0 and later include some or all of the following switches. You can use one to increase the size of the queue and another to bypass the initial prompt, instead specifying the printer's logical name on the command line. Other switches let you fine-tune the amount of time that your computer takes for printing.

SWITCH	FUNCTION
/d:*dev*	Lets you specify the name of the port to which the printer is attached without waiting for a prompt. This switch must appear immediately after the Print command, before any other switches or file names.
/b:*s*	Lets you specify the size of the print buffer. The default size is 512 bytes. A larger buffer—for example, 2048 bytes—is considerably more efficient, but occupies more of your RAM.
/q:*f*	Lets you specify the number of files the queue can hold, from 1 to 32. The default is still 10, as it was in DOS 2.
/m:*m*	Lets you specify the maximum number of clock ticks Print may use to attempt to get a character to the printer before returning control to the program in the foreground. The default is 2, out of 18 ticks per second. Acceptable values range from 1 to 255.
/U:*b*	Lets you specify how many ticks the computer should wait until the printer is available (not busy) before giving up its time to the foreground program. As with the /m switch, the default is 2 and the range is from 1 to 255.
/S:*t*	Lets you specify the time slice allotted to the printer. This actually determines the maximum number of characters the printer will print before returning control to the foreground program. It overrides the amount of time set by the /m switch if that amount of time is not used up.

Warnings The following list summarizes some of the more serious limitations of the Print command:

- You can print only from disk.
- If your disk files are not formatted, the text sent to the printer will be unformatted.
- You cannot remove the disk from which the Print command is printing without losing data.
- You lose access to your printer for other printing tasks, such as dumping screens and listing programs, until the queue is empty.

In addition, the Print command has two other dangers:

- You can lock up your system by telling Print to print to a device that isn't there or isn't turned on. You may have to turn your computer off and on again before you can use it.

- The Print command can crash the system if it needs to access a disk at precisely the same moment as the foreground program.

Tips

If you have any kind of memory board in your computer, it probably came with software that included a print spooler. A print spooler establishes a memory buffer that intercepts anything going to the printer and holds it until the printer is ready to receive it. Some print spoolers also allow you to control many aspects of the printing, including changing the print style as you print, or setting the printer to skip over perforations. Some can establish their buffers in expanded or extended memory (see Memory). If you have such software, try it out before you decide to experiment with the Print command.

PRINTER.SYS

Syntax

```
device=printer.sys lptn=(type[,charset1][,charset2]...
  [,charsetn][,n,])
```

DOS Versions

MS-DOS 3.3 to 4.0
PC-DOS 3.3
IBM DOS 4.0

Overview

If you use DOS 3.3 or later and you have an IBM Proprinter II, Model 4201, or an IBM Quietwriter III, Model 5202, you can configure it to print characters in one or more of the alternative language character sets supplied with DOS. If you use DOS 4.0 you can do the same with an IBM Proprinter XL, Model 4202, an IBM Proprinter X24, Model 4207, or an IBM Proprinter XL24, Model 4208. PRINTER.SYS is the device driver used to configure your printer to print them. You load it with a Device = command in your CONFIG.SYS file. This device driver is useless with any other type of printer.

Procedure

If you have one of the types of printers mentioned above, and you want to print national language character sets other than the default, include these two commands in your CONFIG.SYS file, in the following order:

```
device=ansi.sys
device=printer.sys lptn=(type[,charset1][,charset2]...[,charsetn][,n,])
```

The parameters for PRINTER.SYS are as follows:

- `lpt`*n* is the parallel port to which the printer is attached (see Ports, Serial and Parallel). You may substitute PRN if the port is LPT1.
- *type* is the model number of the printer. Use 4201 for the Proprinter II and Proprinter XL, 5202 for the Quietwriter III, or 4208 for the Proprinter X24 or XL24.
- *charset* is the code number for the character set built into the printer's character ROM chip. In the United States this is 437; in most other countries it is 850, the international character set. You can add as many character sets as your the character ROM chip can print.
- *n* is the number of additional character sets you want to use. This depends on what is built into your character ROM chip. See your printer manual to determine what is available to you.

Examples

To use the international character set on a Proprinter II attached to the second parallel port and still be able to use the default character set, enter this command in your CONFIG.SYS file:

```
device=printer.sys lpt2=(4201,850,437,2)
```

Tips

For complete information on teaching your computer to speak another language, see also ChCP, Country=, COUNTRY.SYS, DISPLAY.SYS, GrafTabl, Keyb, Mode and NLSFunc. Mode includes a table of character-set codes.

PRN *or* PRT

Overview

PRN (or PRT, in some versions of DOS) is an alternative name for LPT1. If you want to redirect printer output to the serial port, directing output to the PRN device will then send printer output to the serial port (see Mode). See Ports, Serial and Parallel, and Device Names for further information.

Prompt

Syntax

{1} `prompt` *`text`*
{2} `prompt`

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Prompt command allows you to customize the DOS prompt shown on your screen. If you do not enter a Prompt command, your prompt will default to A> if you boot from a diskette, or C> if you boot from a hard disk. This prompt tells you the name of the current drive and allows space for you to enter a command.

You can add any text you wish to a prompt. If the text you want to use includes DOS command characters, you can use special commands, called *meta-strings*, to enter these characters (see Command Characters). If you use the ANSI.SYS device driver you can use any of the ANSI escape sequences as part of your prompt (see ANSI.SYS). On a color system, you can use these sequences in your prompt to control the color of your screen display and even make the prompt a color different from that of the rest of the screen.

In addition to text characters, your prompt can display the following system variables:

- The current drive
- The current path

- The current date
- The current time, as of the last time the prompt appeared on the screen
- The name and version number of the DOS you are using

These values are also entered with meta-strings. An additional meta-string allows you to create a prompt that displays more than a single line. Table 22 lists the meta-strings and their effects.

Procedure

{1} `prompt` *text*

To enter a prompt of your choice, enter the Prompt command, followed by the text and meta-strings that will produce the desired result. You should include your Prompt command in your AUTOEXEC.BAT file right after your Path command (see AUTOEXE.BAT File). You can enter Prompt commands at the DOS prompt as well. For example, to display the drive name, path name, and the > character, enter this command:

```
prompt $p$g
```

TABLE 22 **Prompt Meta-Strings**

META-STRING	EFFECT
$t	Displays the current time
$d	Displays the current date
$n	Displays the current drive name
$p	Displays the current drive and pathname
$v	Displays the DOS version name and number
$_	Inserts a carriage-return/line-feed
$b	Inserts a |character
$e	Inserts an Escape acter
$g	Inserts a > character
$h	Backspaces one character
$l	Inserts a < character
$q	Inserts an = character

When your current directory is C:\UTILITY and you have entered this command, your screen displays the prompt

```
C:\UTILITY>
```

{2} `prompt`

To restore the default prompt or create one if something causes your prompt to disappear, enter the Prompt command with no parameters.

Examples

For any system that uses subdirectories, the most basic prompt should be the one generated by pg, described above. Without a display of the path name, you must actually look at a directory or issue a CD command if you forget where you are.

You can add useful information to this prompt. For example, to display the date on one line, and the drive and directory (followed by the > character) on the next, use this Prompt command:

```
prompt $d$_$p$g
```

To this prompt you can add messages on separate lines. For example, you might be able to keep unauthorized users away from your computer with a command such as this:

```
prompt This computer property of Amalgamated Nationwide
  Corp.$_Unauthorized use prohibited.$_$d$_$b$p$b
```

This command would result in the prompt

```
This computer property of Amalgamated Nationwide Corp.
Unauthorized use prohibited.
Tue 6-20-1989
¦C:\REPORTS¦
```

Although this command is shown on more than one line in this book, it must be entered on a single line.

The $h meta-string is useful if you want to display the time to the nearest hour and minute, leaving out the nearest hundredth of a second. The command

```
prompt $t$h$h$h$h$h$h [$p]
```

might display a prompt such as

```
15:42 [C:\NOTES]
```

depending, of course, on the current time and the current directory. Without the $h meta-strings, you would see this:

```
15:42:27.92 [C:\NOTES]
```

You can add an additional message to be even clearer:

```
prompt The time is $t$h$h$h$h$h$h$_[$p]
```

This Prompt command displays

```
The time is 15:42
[C:\NOTES]
```

Notice that space characters in the command are echoed one-for-one as space characters in the prompt.

The $e meta-string is an Escape character, which you can use to enter ANSI escape sequences. Most useful are escape sequences to move the cursor or to display colors. For example, the command

```
prompt $e[31;47m$e[s$e[f $v $e[5C $d $[u$p$g$e[1;33;42m
```

uses ANSI escape sequences for the following:

- Coloring the prompt information red on white (`$e[31;47m`)
- Saving the current cursor position (`$e[s`)
- Moving the cursor to upper-left corner of the screen (`$e[f`)
- Moving the cursor five spaces forward (`$e[5C`)
- Returning the cursor to the position it occupied when it was saved (`$e[u`)
- Coloring the screen green, with bright yellow text (`$e[1;33;42m`)

These additional meta-strings display the DOS version and number ($v) while the cursor is at the upper-left corner, the date ($d) five spaces after the version information, and the drive and path name in brackets at the point you would expect them to appear, after the cursor position has been restored. For legibility, the white stripes displaying the version information and date include a space at each end. For full details, see ANSI.SYS, or the tables of ANSI escape sequences in Appendix A.

Warnings

If you include a prompt as long and complex as some of those described in Examples, and you also use a long Path command, you may see the message

```
Out of environment space
```

When you do, it means DOS will not remember part of your path or prompt. To remedy this situation, expand your environment space with the Shell= or Command commands (see Shell= and Command).

You cannot use ANSI escape sequences in your prompt unless you have loaded ANSI.SYS in your CONFIG.SYS file (see ANSI.SYS and CONFIG.SYS File).

Tips

LOGON.BAT (in the Call entry) and TSR.BAT (in the ANSI.SYS entry) include examples of special-purpose prompts. (See Directories and ANSI.SYS.)

You can use $e meta-strings in Prompt commands in batch files to execute ANSI escape sequences without worrying about how to enter the Escape character. Just remember to enter a Prompt command that sets your prompt back to normal after the other Prompt commands have been executed.

RAMDRIVE.SYS

Syntax

```
device=ramdrive.sys [size] [sector-size] [dir-size] [/switch]
```

DOS Versions

Some MS-DOS versions 2.11 to 4.0

Overview

RAMDRIVE.SYS is a device driver that creates a *RAM disk*—a portion of RAM that is used as an extra disk drive. It is loaded by means of a Device= command in the CONFIG.SYS file (see CONFIG.SYS File). This driver allows you to specify the total size of the disk, the sector size, and the number of directory entries. If you have either PC AT with extended memory (memory beyond 640K), or a 386 PC, you can place the RAM disk in extended memory using the /e switch. If you have an expanded memory board, you can place the disk in expanded memory using the /a switch. The device driver itself uses only about 1K of system memory. (See Memory for a discussion of these types of memory.)

For a complete discussion of RAM disks, including their purposes and uses, see VDISK.SYS and AUTOEXEC.BAT.

Procedure

To use RAMDRIVE.SYS to create a RAM disk, add a line of the following form to your CONFIG.SYS file:

```
device=ramdrive.sys [size] [sector-size] [dir-size] [/switch]
```

The values to be entered in this command are summarized in Table 23.

If your disk is in conventional memory, you can specify a size up to 64K less than your total conventional memory. (If you have 64K or less of conventional memory, you cannot install RAMDRIVE.)

In addition to sector sizes of 128 and 1024 bytes, you can also have sectors of 256 or 512 bytes. The larger the sector size, the more data can be read into memory at a time (see Disk Sectors).

dir-size refers to the maximum number of entries in the root directory. Despite the fact that directory entries are allotted in sectors, and the number is rounded to the nearest sector boundary, you can actually limit the root directory to two entries. Each directory entry takes 32 bytes (see Directories). The actual number of entries you can use will be one less than the number you allocate as *dir-size* because DOS places a volume label on the disk, which uses one entry (see Label).

To create more than one RAM disk, add additional `device=ramdrive.sys` entries to your CONFIG.SYS entry. Each RAM disk loaded will receive the next available drive name. If you have a hard disk, the first RAM disk will be drive D, the next drive E, and so on. If you do not have a hard disk, the first RAM disk will be drive C, and so on. Even if you have only one drive, your first RAM disk will be drive C. A and B are drive names reserved for diskette drives.

When DOS loads the device driver, it displays a message of the following form:

```
RAMDRIVE Version 3.xx virtual disk C:

Buffer size:        64 KB
Sector size:       128
Directory entries:  64
```

The entries shown are for the default values on a system with no hard disk.

To place the RAM disk in PC AT or 80386 extended memory, end the command with the /e switch. To place the RAM disk on an expanded memory board, use the /a switch. Do not use both switches in the same command.

TABLE 23 **RAMDRIVE.SYS Parameter Values**

PARAMETER	UNIT	MINIMUM	MAXIMUM	DEFAULT
size	kilobytes	16	32,768	64
sector-size	bytes	128	1024	128
dir-size	entries	2	1024	64

Examples

See VDISK.SYS for examples. All procedures are the same, except that you substitute the file name RAMDRIVE.SYS for VDISK.SYS. The only exception is that RAMDRIVE.SYS can create RAM disks in expanded memory while versions of VDISK.SYS before 4.0 cannot. For example, the command

```
device=ramdrive.sys 640 1024 32 /a
```

creates a 640K RAM disk in expanded memory, with 1K sectors, and a root directory with room for thirty-one entries besides the volume label.

Warnings

Anything on a RAM disk disappears forever when the power is off, even for a moment. It is therefore very important to back up your files frequently from a RAM disk to a less volatile medium.

The defaults described in this entry are those used by generic MS-DOS 3.2. If you have a another version of DOS, its parameters may have different defaults or accept different ranges of values. Check your DOS manual for complete information.

Do not use the DOS RAMDRIVE.SYS with Microsoft Windows. With that program, use only the version of RAMDRIVE.SYS included in the package.

Tips

For full information on how and why to use RAM disks, see AUTOEXEC.BAT File and VDISK.SYS. The latter entry also shows sample programs for starting applications on RAM disks and backing up data from them. The former gives full information on setting up RAM disks once they are configured.

RD

Syntax

{1} rd *path*
rd*path*

{2} rd *drive path*

DOS Versions

{all} MS-DOS 2.1 to 4.0
{all} PC-DOS 2.0 to 3.3
{all} IBM DOS 4.0

Overview

On a disk with tree-structured directories, the RD command deletes a subdirectory. RD stands for remove directory; the longer form RmDir is equivalent.

You may want to delete a subdirectory under these circumstances:

- When you replace an old software package on your hard disk with a new one
- When you have directories full of files that you no longer use
- When you want to change the logical organization of your disk

If your directory tree helps you find the files you need, it can save you time; if it doesn't save you time, it can impair your productivity. Directories that are not logically organized or that are too full impede your progress. Thus, you may need to remove subdirectories periodically and make new ones to make your tree more efficient.

There is no set rule about the best organization; you must set up a structure that you can work with comfortably (see Directories).

Procedure

In the discussion that follows, assume that the hard disk is drive C. The directory structure will follow Figure 28 in the Directories entry. In the following examples, you may want to use Figure 28 for reference.

{1} rd *path*
rd*path*

To delete a subdirectory on the current drive, enter the RD command followed by the name of the directory to delete. To delete a subdirectory that is a child of another subdirectory, specify the path to the directory you want to delete.

Like all commands using path names, RD can use either absolute or relative path names. You needn't enter a space between the command and the path name when using an absolute path name.

{**2**} `rd` *`drive path`*

To delete a directory on a disk that is not current, include the drive name in the command, before the path name.

For example, to delete the QTR1 subdirectory from the \REPORTS\SOUTH directory, use the command

```
rd\reports\south\qtr1
```

If the \REPORTS\SOUTH directory is current, you can use a relative path name:

```
rd qtr1
```

Warnings There are three circumstances under which you cannot delete a subdirectory:

- When it is current
- When it contains files
- When it has children

If any of these conditions is true, you will see the message

```
Invalid path, not directory,
or directory not empty
```

If the directory is current, use the CD command to change to another directory, then make appropriate changes to the path name in the RD command. If the directory contains files, erase them. You can do so by

issuing a command referring to the directory. For example, to erase all the files from the QTR1 subdirectory of \REPORTS\SOUTH, you could enter

```
del \reports\south\qtr1
```

If the directory you want to delete has children, you must first erase the files from them and delete those directories. To delete the \REPORTS\SOUTH directory, for example, you would first have to issue the commands

```
del \reports\south\qtr1
rd\reports\south\qtr1
del \reports\south\qtr2
rd\reports\south\qtr2
del \reports\south\qtr3
rd\reports\south\qtr3
```

If you have taken all these steps, and you still see the message, there are hidden files in the directory (see File Attributes). You should be aware that some copy-protected software creates hidden files in various directories. Also, some utility software occasionally goes haywire and accidentally changes the hidden or system attribute on a file. Unfortunately, DOS provides no means of addressing the hidden or system attributes. You must use some utility software that does address these attributes to expose the hidden files or use DEBUG to change the value of the attribute byte (see DEBUG and File Attributes). If the files were created by copy-protected software, use the software's uninstall procedure before attempting to remove the software from your disk.

When an RD command includes a drive name, you must enter a space between the command and the parameter, but you cannot have a space between the drive name and the path name. The following forms would result in error messages:

```
rdc:\reports\midwest\qtr4
rd c: \reports\midwest\qtr4
```

Recover

Syntax

{1} recover [*drive*][*path*\]*filename.ext*
{2} recover [*drive*]

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Recover command recovers files that have been written to bad sectors. (See Disk Sectors and ChkDsk for information on bad sectors.) It can also recover some or all of your files when the file allocation becomes damaged.

Procedure

Recover is an external command; using it requires having the DOS program RECOVER.COM (or RECOVER.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:recover b:datafile.txt
```

If you use DOS 3.0 or later, and the program is not in the current directory or on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\recover a:
```

Recovering a File with Bad Sectors

{1} recover [*drive*][*path*\]*filename.ext*

To recover a file that has been written to bad disk sectors, enter the command

```
recover filename.ext
```

where *filename.ext* is the file that includes the bad sectors. If the file is not in the current directory, preface it with the name of the drive and/or directory where the file is located. If you use wildcard characters in the file name, DOS will recover only the first file in the directory whose name matches the wildcard pattern (see Wildcard Characters).

DOS then reads the file, one sector at a time, skipping over the bad sectors and marking them as unusable. It then writes the file elsewhere on the disk. You should immediately open the file in the application that you used to create it. Some of the file will probably be missing because the part that was in the bad sectors will be excluded.

If the file was created by a program that includes internal pointers or formatting information in its data files, the recovered file will probably still be unusable. The most likely way to recover the data from it is to load it into DEBUG or, better yet, a utility that can display the text in a file without extraneous characters, then dump the result to the printer or an ASCII text file. You will then have to attempt to reconstruct the file from your data, which is no fun. Avoid this scenario by making frequent backup copies on another disk.

Recovering a Disk with a Bad File Allocation Table

{2} `recover [drive]`

When DOS reads a file, it uses the file allocation table in conjunction with information in the directories to locate all of the file's clusters (see File Allocation Table, Directories, and Disk Clusters). The file allocation table is so important that every disk has two of them. If something goes wrong with one, DOS uses the other to correct it. Sometimes, however, both copies of a file allocation table become damaged. If this happens, you will see a message saying `Sector not found` or `File allocation table bad` when you try to read from a disk. Running ChkDsk at such a time doesn't help, because ChkDsk also uses the file allocation table.

Even if the file allocation table is unreadable, the files may still be intact.

However, this won't help you if you can't find the files. This is the only point at which you should use the Recover command without a file name. Enter the command `recover` specifying the name of the drive containing the damaged disk if it is not in the current drive. DOS will load the RECOVER program and display a message of the form

```
Press any key to begin recovery of the
file(s) on drive drive
```

When you see this message, replace the disk with the RECOVER program on it with the damaged disk if you need to load the program from the same drive.

Some especially friendly versions of DOS will also display the message

```
WARNING! All files on drive drive will be renamed and your
directory structure destroyed. Do you want to continue the
recovery (Y/N)?
```

This is precisely what will happen, even if your DOS doesn't tell you so. DOS will read whatever it takes to be a file, then write each one to a new file with a name of the form

```
FILEnnnn.REC
```

where *nnnn* is a number from 0000 to 9999. All of these files will appear in the root directory. If the root directory becomes full, the recovery process will stop.

If you're not sure what the files contain, and you don't want to examine them immediately, copy them to another disk, erase them from the damaged disk, then issue the Recover command again.

You must delete the files to make room for the names of other files. Repeat the process until all files have been recovered. Each time you enter the Recover command, the maximum number of files that can fit in the directory will be recovered, even if they have already been recovered. In other words, the file numbers will again begin with 0000.

Now you have a mess on your hands. If your files were not too badly fragmented the recovered files may actually be whole files. (See ChkDsk and File Allocation Table for information on file fragmentation.) This is a big help since the only clue you have about the original identity of the files is their size as listed in the directory.

If the files were indeed fragmented, each fragment will be written to a separate file. Moreover, every file's length will be a multiple of a disk sector, so you can only guess their original length.

If the files were program files, the next best step is to create a new working disk from the original distribution disk. If they were text files, you can examine them with the Type command or a word processor and try to put them back together. Needless to say, you should use the Recover command only in extreme circumstances.

Warnings

Recover is a dangerous command that you won't need if you back up all your files regularly. However, if you must use this command, *always* follow it with a file name unless your disk is completely unreadable.

Redirection and Pipes

Syntax

{1} *program>destination*
{2} *program<input*
{3} *program>>textfile.ext*
{4} *program|filter*[*>destination*]

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 1.0 to 3.3
{**all**} IBM DOS 4.0

Overview

DOS assumes the keyboard as the *standard input* device and the screen as the *standard output* device. Together, these devices form the *console*, which has the reserved name CON (see Device Names). You can use *redirection* to force some programs to accept input from some point other than the standard input (usually a file). You can also redirect output from most programs to a device other than the console or to a file.

Additionally, you can pass the output from one program through another program before it reaches its destination. The intermediate program, which changes the data in some way, is called a *filter* (see Filters). When you use a filter, your data passes through a *pipe*.

The symbols >, >>, and < are *redirection operators* that tell DOS where you want to send your data. The symbol | is the *pipe operator*, which separates one program in a pipe from another. In a command line, spaces preceding or following these symbols are optional.

Procedure

Redirecting Output

{1} *program>destination*

To redirect output, enter the name of the program or command that will produce the output, followed by a greater-than symbol, followed by the destination to which you want to send the output. The destination can be a DOS device or a file.

Redirecting Input

{2} *program<input*

To redirect input, enter the name of the program to execute, followed by a less-than symbol, followed by the name of the file from which the input will be taken. As a rule, there are few circumstances in which you can redirect input. However, some exceptions follow.

Appending a File

{3} *program>>textfile.ext*

To append information to an existing text file, enter the name of the program or command that will produce the information, two greater-than symbols, and the name of the file to which the information is to be appended. If the file does not exist, DOS will create it. If it does exist, the new information will appear immediately after the last character in the existing file.

If the last character in the existing file is an end-of-file mark (Ctrl-Z), you will be unable to see the new information when you use the Type command to view the file. If your word processor ends its files with a Ctrl-Z, you will not see the new information in your word processor either. To see the information, you will need to load the program into DEBUG and replace the Ctrl-Z (hex 1A) with a space (hex 20) (see DEBUG).

Using a Pipe

{4} *program|filter*[*|filter*][*>destination*]

To pass data through a pipe, enter the name of the program or command that will produce the data, followed by a | symbol and the name of the filter that will alter the data. If you do not specify a destination, the output will automatically go to the console (that is, the screen). You may want to use this technique to print out the READ.ME files that come with many software packages.

You can direct the output of most DOS commands and many other programs to a file instead of to the screen. To create a disk file containing the complete path names of all the files on your hard disk, enter `chkdsk /v>catalog.hd`. Similarly, to print a directory with a single command, enter `dir>prn`.

Examples

Redirecting Input

{2} *program<input*

Suppose you want to format a stack of diskettes. When you enter the Format command, DOS first asks you to place a diskette in the target drive. Next, it formats the diskette, then asks if you want to format another. If you type Y, the cycle starts over. You have to watch your screen or your drive light because the program doesn't beep when it's time to respond.

The program MFORMAT.BAT (for Mass Format), which follows, reduces the number of times you must respond, and adds a beep so that you know when your attention is required. MFORMAT.BAT uses a file called MFORMAT as input, which contains the two characters

```
.n
```

The period is accepted as input when the

```
Strike a key when ready . . .
```

message appears. The n is accepted as input when the

```
Format another (Y/N)?
```

message appears.

MFORMAT.BAT, Version 1

PROGRAM	COMMENTS
`format b:<mformat`	Formats the diskette; accepts keystrokes from the file MFORMAT.
`pause ^G`	Prompts for a keystroke with a beep.
`%0`	Calls MFORMAT.BAT.

The DOS filter program More (see More) stops screen output from scrolling when a full screen of text has appeared. It displays the message --More-- at the lower-left corner of the screen. One way to employ this message is to use it in place of the Type command and redirect the file you want to look at to it:

```
more<textfile.ext
```

A third use for redirected input is sending a *script file* to DEBUG. See FRIDAY.DEB in the ErrorLevel entry for an example of such a script file.

Appending a File

{3} *program>>textfile.ext*

To create a catalog of all your diskette directories, place each diskette in drive A, and enter the command

```
dir a:>>c:catalog.fd
```

If the file CATALOG.FD does not exist when you enter the command, DOS will create it. If it does exist, the directory of the new diskette will be appended to the file. As you repeat the command with new diskettes, each new directory will appear after the end of the previous directory.

Using a Pipe

{**4**} *program*¦*filter*[¦*filter*][>*destination*]

You can use a pipe to accomplish the same thing you used redirection for in MFORMAT.BAT, version 1, discussed under Redirecting Input. In version 2 of MFORMAT.BAT, the Echo command sends characters to the Format command by way of a pipe; otherwise, the program works exactly the same as version 1. The advantage of using redirected input is that it avoids creating the temporary file, thereby saving time. The advantage of a pipe is that you can avoid cluttering up your disk with tiny input files.

MFORMAT.BAT, Version 2

PROGRAM	COMMENTS
`echo .N¦format b:`	Formats the diskette. Keystrokes are sent to the command by the Echo statement.
`pause ^G`	Prompts for a keystroke with a beep.
`%0`	Calls MFORMAT.BAT.

The lines appearing in SETDATE.BAT, when included in an AUTOEXEC.BAT file, will place the date and time in the environment (see Environment Space). Once they are in the environment, they can be read by other batch files that need to know the date and time.

SETDATE.BAT follows. You also need two other files, TODAY and NOW, for the process to work properly.

SETDATE.BAT

PROGRAM	COMMENTS
`echo¦more¦date>todayis.bat`	Creates TODAYIS.BAT, with the current date
`copy today current.bat`	Creates the version of CURRENT.BAT that places the date in the environment
`call todayis`	Executes TODAYIS.BAT, to create the environment variable TODAY with the current date
`echo¦more¦time>nowis.bat`	Creates NOWIS.BAT, with the current time
`copy now current.bat`	Creates the version of CURRENT.BAT that places the time in the environment
`call nowis`	Executes NOWIS.BAT, to create the environment variable NOW with the current time

The logic of this procedure is to place the date and time into batch files, and then have those batch files copy the information to the environment by using the Set command (see Set). Basically, what you want to do is to get the output of the Date and Time commands into a batch file. The command `echo¦date>todayis.bat` won't work, because it puts the text `date` into a batch file. When you execute this file, this command will simply issue the Date command, just as if you had entered it at a DOS prompt.

The command `date>todayis.bat` will, in fact, echo the output of the date command into the batch file. However, your computer will hang while waiting in vain for the Echo command to enter a response to the

```
Enter new date (mm-dd-yy):
```

prompt, which is already in the batch file. You can solve this problem by adding the More filter to the pipe:

```
echo¦more¦date>todayis.bat
```

When used this way, the More filter appends a carriage return to the output of the Date command, because the Date command doesn't fill the screen. The resulting file, TODAYIS.BAT, now contains text such as

```
Current date is Thu 7-13-1989
Enter new date (mm-dd-yy):
```

When you execute TODAYIS.BAT, it immediately tries to execute the command Current, which you must create since it doesn't exist. Create a file called CURRENT.BAT containing the command

```
set today=%4
```

This takes the fourth parameter to CURRENT (the actual date) and assigns it to the environment variable TODAY.

If you want only the date in the environment, you're now finished. If you also want the time in the environment, you have a minor problem, but you can use the same technique to create the batch file NOWIS.BAT:

```
echo|more|time>nowis.bat
```

The resulting file will contain text such as

```
Current time is 1:23:34.45
Enter new time:
```

Now when you execute NOWIS.BAT, it will again try to execute the command Current. But CURRENT.BAT doesn't contain the commands to place the time in the environment. To do that, CURRENT.BAT must read

```
set now=%3
```

Thus the construction of SETDATE.BAT. The version of CURRENT.BAT for the date is stored in a file called TODAY, and that for the time in one called NOW. Before the files that execute CURRENT.BAT are invoked, the proper text file is copied to CURRENT.BAT.

Note that this technique works only with DOS 3.3 or later. You cannot substitute `command /c` for `call` because the secondary command processor that `command /c` loads creates its own environment (see Command). When you exit to TODAYIS.BAT, the environment containing the variable TODAY will disappear. The same will happen with NOWIS.BAT.

LOGON.BAT, in the Call entry, makes use of these files to log computer use and also uses redirection to append data to the log file. HDBACKUP.BAT (in the Hard Disks, Backing Up and Restoring, entry) uses the same techniques to ensure timely hard disk backups.

Warnings

Redirected input cannot supply command-line arguments to a command. Input also cannot be redirected into a batch file to supply parameters to replaceable parameter symbols (see Batch Files).

Redirecting input is always risky. If you don't set up your input correctly, control may not be returned to the console, and you will have to reboot.

Tips

For related information, see Ctty, Device Names, and Filters.

Rem

Syntax

{**1**} `rem` *text*
{**2**} `rem`

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Rem command adds remarks or comments to a batch file. In DOS 4.0, it can also be used to add comments to a CONFIG.SYS file (see CONFIG.SYS File). Adding comments to programs is useful for two reasons. First, it explains your code to others who may use it. Second, it helps you remember why you wrote a particular segment of code the way you did. (Six months after writing a program, you may have forgotten how the program works. If the code includes an explanation, it will be much easier to modify the code if you need to.) Remarks can also be used to display text in a batch file. If Echo is on, all the text in a batch file appears on the screen as the program executes, except for labels (see Echo and GoTo).

In DOS 2, commands in Rem statements are executed. However, commands preceded by colons are not. Thus, you can set up remarks containing commands as if they were labels (see GoTo).

In DOS 3.0 and later, you can prevent a command in a Rem statement from being executed by placing it in quotation marks. For an example, see ACCOUNTS.BAT, in the GoTo entry.

Procedure

{**1**} `rem` *text*

To enter a remark, use the command Rem, followed by the text of a remark:

```
Rem This is the text of a remark
```

{2} rem

If Echo is on, you can use Rem statements containing only the Rem command to create blank lines on the screen. Examples appear in the DATABAK.BAT program and in Figure 80.

Examples

In a book, programs can be annotated in a parallel column. However, you must use Rem statements to include annotations in your own batch programs. The following program is a version of BUSLTR.BAT, the simplest version of which appears in the Batch Files entry. The following version includes remarks in place of the annotation column that appears in the simplest version.

BUSLTR.BAT with Remarks

```
echo off
rem Load the SuperKey macro file for WordStar
key e:ws /ml
rem Initialize the mouse driver, do not display messages
mousesys >nul
rem Load the mouse menu for WordStar
m_ws
rem Make the \LETTERS\BUSINESS directory current
cd\letters\business rem
Load WordStar, open the file named on the command line
ws %1.ltr
rem Clear the SuperKey macro file
key /mc
rem Deletes the backup copy of the letter
del %1.bak
rem Display the directory
dir /p
rem re-initialize the mouse driver, do not display messages
mousesys >nul
rem Load the default mouse menu
m_plain
rem Make the root directory current
cd\
```

```
D:\>databak

D:\>c:

C:\>cd\dos

C:\DOS>rem Any data on the diskette in drive B will be lost

C:\DOS>format b:
Insert new diskette for drive B:
and strike ENTER when ready

Format complete

    362496 bytes total disk space
    362496 bytes available on disk

Format another (Y/N)?n

C:\DOS>rem Verifying the integrity of the diskette in drive B.

C:\DOS>rem

C:\DOS>rem If you see a message regarding bad sectors,

C:\DOS>rem press Ctrl-C and start over.

C:\DOS>chkdsk b:

   362496 bytes total disk space
    21504 bytes in 8 user files
   340992 bytes available on disk

   655360 bytes total memory
   444880 bytes free


C:\DOS>rem Copying files from drive A to drive B

C:\DOS>copy a:*.* b: /v
A:TOTALS.DAT
A:ADDRESS.MCD
A:MAX_FIGS.FW2
A:CA611.LTR
A:CA910.LTR
A:CULLINAN.LTR
A:MCD1013.LTR
A:SUBMIT.1
        8 File(s) copied

C:\DOS>dir b:

 Volume in drive B has no label
 Directory of  B:\

TOTALS   DAT      256  10-13-87   2:14p
ADDRESS  MCD      256  10-13-87   2:15p
MAX_FIGS FW2     7696   9-14-87  10:52a
CA611    LTR      768   6-11-87   1:05p
CA910    LTR     1920   9-10-87   2:15p
CULLINAN LTR     3072   9-22-87   2:16p
MCD1013  LTR     3072  10-13-87   2:04p
SUBMIT   1       1476  10-13-87   2:44p
        8 File(s)    340992 bytes free

C:\DOS>_
```

Figure 80: Output from DATABAK.BAT with remarks

A simple batch file like this one, which does nothing more than execute a series of commands in order, doesn't need extensive remarks. However, when your files include loops, or GoTo statements, remarks are helpful for explaining the flow of the program. See GoTo, and Hard Disks, Backing Up and Restoring, for examples of programs using GoTo statements.

The following program shows a version of DATABAK.BAT that uses remarks to explain the program to the user as it executes. This is a variation on the program that appears in the Batch Files entry. Figure 80 shows the display that results from executing the program using MS-DOS 3.2. Both the Rem command and the prompt appear each time a Rem statement is executed.

A blank line precedes each prompt. The resulting display is neither attractive nor easy to read.

DATABAK.BAT with Remarks

```
c:
cd\dos
rem Any data on the diskette in drive B will be lost
format b:
rem Verifying the integrity of the diskette in drive B.
rem
rem If you see a message regarding bad sectors,
rem press Ctrl-C and start over.
chkdsk b:
rem Copying files from drive A to drive B
copy a:*.* b: /v
dir b:
```

Tips

To include remarks in a batch file that will never appear on the screen during execution, enter them as labels; that is, precede them with a colon instead of Rem. Be sure that the first eight characters of the remark are not the same as any of the functioning labels in your batch file.

A much more elegant way of displaying messages while executing a batch file is to turn Echo off and place your messages in Echo statements. For an example, see the version of DATABAK.BAT in the Echo entry.

For related information, see AUTOEXEC.BAT File, Batch Files, Echo, Pause, and Type.

Ren

Syntax

ren [*drive*][*path*\]*oldname.ext newname.ext*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Ren command is used to rename files, individually or in groups.

Procedure/ Examples

To rename a file, enter the Ren command followed by both the current name of the file and the new name for the file. For example,

```
ren thisfile.doc thatfile.doc
```

changes the name of a file called THISFILE.DOC to THATFILE.DOC. If the file is not on the current drive and directory, preface the original file name with a drive and/or path name. For example, if THISFILE.DOC is on a directory called \MATERIAL on drive C, and some other drive is current, enter the command

```
ren c:\material\thisfile.doc thatfile.doc
```

To rename groups of files, use wildcard patterns (see Wildcard Characters). To change the extension of all the .BAK files on the current directory to .OLD, enter

```
ren *.bak *.old
```

If you have a file named HORSE.TXT, the command

```
ren horse.* m?u??.*
```

renames the file to MOUSE.TXT.

If the file name that would result from a Ren command is a name of a file that already exists in the directory, you will see the message

```
Duplicate file name or File not found
```

You will also see this message if the file you are attempting to rename does not exist or if you mistyped its name.

Warnings

You cannot use wildcard patterns to add characters to a group of file names. For example, suppose you have files named

TXT1.DOC
TXT2.DOC
TXTADD.DOC

If you enter the command

```
ren txt?.doc txta?.doc
```

you might expect TXT1.DOC to become TXTA1.DOC, and TXT2.DOC to become TXTA2.DOC. In fact, all that happens is that the message

```
Duplicate file name or File not found
```

appears.

Entering the command

```
ren txt*.doc text*.doc
```

will not result in files named

TEXT1.DOC
TEXT2.DOC
TEXTADD.DOC

Instead, TXT1.DOC would become TEXT.DOC, and the other two would not change. You would see the message

```
Duplicate file name or File not found
```

informing you that DOS was trying to rename TXT2.DOC to TEXT.DOC, which already exists.

Tips

You can rename files while copying them with the Copy, Replace, and Xcopy commands by specifying a file name as part of the target. See the respective commands for details.

Rename. *See* Ren

Rename is an alternative name for Ren.

Replace

Syntax

replace [*source*]*filename.ext* [*target*][*/switches*]

DOS Versions

MS-DOS 3.2 to 4.0
PC-DOS 3.2 to 3.3
IBM DOS 4.0

Overview

Replace is a selective copy command that allows you to:

- Copy only files whose names already exist on the target
- Copy only files not present on the target
- Select the files to be copied by name
- Select the files to be copied by date
- Select the files to be copied by archive status
- Replace files in subdirectories of a given directory

You can also use Replace to write over read-only files (the Copy command cannot do this). Thus, you can use it to update operating system files, although the Sys command is specifically designed for that purpose (see Sys).

The power of the Replace command lies in its many switches, which give you many ways to select the files to copy. Following are the switches and their effects:

SWITCH	EFFECT
/a	Copies only files that do not exist in the target directory (that is, appends files to the target).
/d	Replaces only files that exist on the target, but that are more recently changed on the source than on the target (available in MS-DOS 3.2 and 3.31 only).
/s	Replaces files of the same name in subdirectories of the target, regardless of their location on the source.

SWITCH	EFFECT
/r	Replaces read-only files. (If read-only files are among those selected on the source and you do not use this switch, Replace will stop copying files and terminate.)
/p	Displays each file name before copying, requesting your permission to continue by displaying a message of the form `REPLACE SOURCE/FILENAME.EXT (Y/N)?` This gives you the opportunity to exclude files that may have matched your criteria but that you don't want to replace on the target.
/w	Displays the message `Press any key to begin replacing file(s)` and waits for a keypress. This gives you the opportunity to switch diskettes in a drive if you have read the Replace program from a diskette and want to use the same diskette drive as either the source or the target.

Procedure

Replace is an external command; using it having the DOS program REPLACE.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\replace *.dat b:
```

You must specify a source file name. You can use wildcard characters in the file name (see Wildcard Characters). If the source drive and/or directory are current you need specify only the source file name and the target:

```
replace *.dat b:
```

If the target drive is current, you must specify the source drive as well as the target:

```
replace c:*.dat b:
```

If the target drive is current, and the source directory is not current on the source drive, specify the source drive and directory:

```
replace c:\accounts\*.dat b:
```

The screen will then display the name of each file on the target that is being replaced:

```
Replacing B:\RECV.DAT
Replacing B:\CHART.DAT
Replacing B:\PAYBL.DAT
Replacing B:\PAYROLL.DAT
Replacing B:\INCOME.DAT
```

and so on.

For examples of programs using the Replace command with switches to replace files selectively, see ACCOUNTS.BAT in the GoTo entry and UPDATE.BAT in the Hard Disks, Backing Up and Restoring, entry.

Warnings

It is very easy to confuse the syntax of Xcopy and Replace if you use both commands frequently. However, these commands are quite different from each other.

With Xcopy, you can specify only the source if the target is current or specify the source as a single period if it is current. With Replace, you may be tempted to treat the source and target the same way. If the source is current you might enter a command of the form

```
replace target
```

For example, if drive D is current, and you want to copy the files to a directory on drive C, you might think of entering the command

```
replace c:\accounts\*.dat
```

However, Replace requires that you always specify a source file name. A command such as this will have the opposite effect from what you intend, replacing the files on drive D, the intended source, with those from drive C, the intended target, wiping out your latest modifications. Heed the voice of experience!

The PC-DOS versions and the MS-DOS 3.3 version of Replace do not have a date switch. The DOS 4.0 version uses /u for a date switch in place of /d.

Tips

If you use an earlier version of DOS that does not include the Replace command, you can use a For . . . In . . . Do loop. The command

```
for %%f in (filename.ext) do if not exist a:%%f copy %%f a:
```

can substitute for `replace /a` if the source is current. Similarly,

```
for %%f in (filename.ext) do if exist a:%%f copy %%f a:
```

can substitute for the Replace command with no switches. For a program using commands of this type, see `UPDATE.BAT` in the Hard Disks, Backing Up and Restoring, entry. See the For . . . In . . . Do entry for an explanation of this line of code. For related information, see Copy and Xcopy.

Restore

Syntax

restore *target* [/*switches*]

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Restore command restores files that have been backed up using the Backup command. It has no other purpose.

You have a variety of choices regarding what to restore with the Restore command. You can restore

- All of a hard disk
- A single subdirectory
- A particular branch of a subdirectory tree
- Files changed or deleted since they were backed up
- Files created or changed before or after a given date (and time)
- Files that are no longer on the hard disk
- Particular files
- Files that have changed since you last did a backup

You determine which files are to be restored by the way you specify the target and by the switches you use. In DOS 3.0 and later, you can also restore

- Diskettes to other diskettes
- Hard disks to other hard disks
- Hard disks to themselves
- Hard disks or diskettes to a network file server

All these options boil down to two fundamental choices: you can restore either an entire hard disk or only part of one. If you restore only part, you

can specify the part using a variety of criteria. Generally, you restore a complete hard disk only if you have reformatted the disk or if you are replacing one hard disk with another. Otherwise, you restore only files that have become corrupted or have been accidentally erased. If you have DOS 3.0 and later, you might restore files from your backup diskettes to other diskettes in order to have usable versions of your files. The main purpose of restoring a file or files, however, is to restore the integrity of your hard disk.

Procedure

Restore is an external command; using it requires having the DOS program RESTORE.COM (or RESTORE.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:restore c:\data\oldfile.txt
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\hdbackup\restore c: /s/p/m
```

The target of a Restore command can be a drive, a directory, a file name, or a combination of these elements. To restore the entire hard disk from a diskette series, use the command

```
restore c:\/s
```

This command tells Restore to restore files on the root directory of drive C, and all its subdirectories. If you do not specify the root, the target will be the current directory of the hard disk. If you do not use the /s (subdirectories) switch, only the current directory will be restored.

To restore particular files, you include file names (and path names if necessary) in the command line. Wildcard characters are permitted.

You must use additional switches if you want to restore files on some other basis. Besides the /s switch, Restore has one additional switch in DOS 3.2 and earlier, the /p switch. This switch asks your permission before restoring each file, thereby allowing you to choose not to restore a particular file.

DOS versions 3.3 and later have six additional switches that allow you to select files for restoration on other bases:

SWITCH	EFFECT
/a:*date*	Restores all files created or changed on or after the specified date. (Use the date format for the country to which your computer has been set; see Country= or Select for more information about selecting date formats.)
/b:*date*	Restores all files created or changed on or before the specified date. (Use the date format for the country to which your computer has been set; see Country= or Select for more information about selecting date formats.)
/e:*hh:mmc*	Restores all files created or changed at or earlier than the specified time. (The time is specified in a twelve-hour format, where *c* is either *a* for AM or *p* for PM. If you have nationalized your computer to a country that uses a different time format, use the time format for that country.)
/l:*hh:mmc*	Restores all files created or changed at or later than the specified time. (The time format is the same as for the /e switch.)
/m	Restores all files that were modified or deleted since they were backed up.
/n	Restores all files that are not on the target disk.

If you use the switches that include dates or times, use the date or time format for your country. To specify files created or changed before or after a particular point in time, use the /a and /l, or the /b and /e switches together. Do not use the /m and /n switches together.

When you enter the command, you will see a message similar to

```
Insert backup diskette 01 in drive a:
Strike any key when ready
```

If you are restoring to another diskette, you will also be asked to insert the target diskette in the target drive and strike a key. When you strike a key, a message of the form

```
*** Files were backed up mm-dd-yy ***
**** Restoring them from drive A: ***
Diskette: nn
```

appears, and the restoration begins. When all matching files from the diskette are restored, you will be prompted to insert the next diskette.

If you insert your backup diskettes out of order, you will see a message such as

```
Warning! Diskette is out of sequence
Replace diskette or continue if okay
Strike any key when ready
```

Examples

Suppose you entered a Restore command such as

```
restore c:\ /1:1:00p /s
```

This command restores all files that have a directory time of 1 PM or later to all subdirectories drive C; the date is insignificant. To restrict the files to those created or modified after August 10, 1989 at 1 PM, use the command

```
restore c:\ /a:8/10/89 /1:1:00p /s
```

Tips

Restore has exit codes that you can test with the ErrorLevel command (see ErrorLevel).

For related information, see Backup; and Hard Disks, Backing Up and Restoring.

RmDir. *See* RD

RmDir is an alternative name for RD.

Select

Syntax

{1} `select [source¦target[\path]] country keyboard`
{2} `select menu`

DOS Versions

{1} Some MS-DOS versions 3.1 to 3.3
{1} PC-DOS 3.0 to 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

The Select command allows you to select a default country code and keyboard layout, format a bootable disk (hard or floppy), and create

AUTOEXEC.BAT and CONFIG.SYS files, all with a single command. It appears in PC-DOS 3.0 and later, and some manufacturers' versions of MS-DOS 3.1 and later.

The command has the same effect in each release of DOS, except in PC-DOS 3.0, where it won't work on a hard disk. The way Select accomplishes its task differs from one release to another.

Procedure

Select is an external command; using it requires having the DOS program SELECT.COM on a diskette in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
a:\dos\select 001 us
```

To select a default country code and keyboard layout and create a bootable disk with all the DOS external programs on it, enter the Select command followed by both the three-digit country code for the country you want to use as a default, and the corresponding two-letter keyboard-layout code. See Table 7 in the Country= entry for country codes and Table 17 in the Keyb entry for a table of keyboard-layout codes.

You can also specify a source drive, which must be a diskette drive. If you specify a target drive, you may also specify the directory in which you want the DOS external programs placed. The Select command creates this directory.

Examples

Select in DOS 3.0 and 3.1 To create a DOS 3.0 or 3.1 system diskette in a two-drive system configured for France, enter the command

```
select 033 fr
```

When you use the PC-DOS 3.0 Select command, the DOS diskette is copied using the DiskCopy command (see DiskCopy). Next, it creates a CONFIG.SYS file (see CONFIG.SYS File) on the copy, consisting of the command

```
COUNTRY=033
```

Finally, it creates the AUTOEXEC.BAT file that follows on the target drive (see AUTOEXEC.BAT File).

Sample AUTOEXEC.BAT File Created by the Select Command

COMMANDS	COMMENTS
PATH \;	Places the root directory on the search path
KEYBFR	Loads the keyboard-mapping table for France
ECHO OFF	Suppresses display of commands
CLS	Clears the screen
DATE	Prompts you for the current date
TIME	Prompts you for the current time
VER	Displays the name of the DOS version.

The DOS 3.1 version of Select uses DiskComp to verify that the copy is accurate before writing the CONFIG.SYS and AUTOEXEC.BAT files.

Select in DOS 3.2 To create a DOS 3.2 system diskette in a system with a single diskette drive configured for Belgium, enter the command

```
select a:¦a: 031 fr
```

(In DOS 3.2, there is no separate keyboard file for Belgium, so the French keyboard is used.)

The DOS 3.2 version of Select does not use DiskCopy; instead, it first formats the target diskette using the Format command (see Format). Next, it creates the AUTOEXEC.BAT and CONFIG.SYS files. Finally, it copies the files from the source using the Xcopy command. Hence, a CONFIG.SYS or AUTOEXEC.BAT file on your source diskette will overwrite those created by the Select command. If these files are not on the source diskette, the CONFIG.SYS file will read

```
country=031
```

and the AUTOEXEC.BAT batch file will be the same as the one already listed.

Select in DOS 3.3 To format a hard disk with DOS 3.3 or later using the select command and to nationalize the computer for Belgium, with the DOS files in their own directory, enter the command

```
select c:\dos 031 be
```

If you have previously formatted the hard disk, you will see the usual warning message that the Format command displays before formatting a

hard disk (see Format). If you have given it a volume label, you will be asked to enter the volume label to verify that you want to erase all of the hard disk you specified (see Label). If the volume label you enter does not match that on the hard disk, the command will terminate. This, too, is a product of the Format command. If you have not previously formatted the hard disk, you will be prompted for a volume label.

The Format command transfers the system files to the root directory of the hard disk and prompts you for a volume label if the disk has none. Next, Select invokes Xcopy to copy the DOS files to the \DOS directory, which it creates. Finally, it creates a CONFIG.SYS file containing only the command

```
COUNTRY=031,850 C:\DOS\COUNTRY.SYS
```

This command does three things:

- The code 031 nationalizes the computer for Belgium.
- The code 850 loads the international font table so that you can display the accented characters. (See Mode for a table of font codes.)
- The path C:\DOS\COUNTRY.SYS tells DOS that the COUNTRY.SYS file is in the \DOS directory of drive C. (COUNTRY.SYS contains the national default tables that the Country= command uses to nationalize the computer. See COUNTRY.SYS and Country= for further details.)

Finally, Select creates the AUTOEXEC.BAT file that follows.

Sample AUTOEXEC.BAT File from DOS 3.3 Select Command

COMMANDS	COMMENTS
`PATH C:\;C:\DOS;`	Places the root directory and the \DOS directory on the search path, leaving room to add other directories.
`KEYB BE 850 C:\DOS\KEYBOARD.SYS`	Loads the keyboard-mapping table for Belgium, with the international font; tells DOS to look in the \DOS directory for KEYBOARD.SYS, which contains the keyboard-mapping tables.

continues

COMMANDS	COMMENTS
ECHO OFF	Suppresses display of commands.
CLS	Clears the screen.
DATE	Prompts you for the current date.
TIME	Prompts you for the current time.
VER	Displays the name of the DOS version.

Select in DOS 4.0 The DOS 4.0 version of Select, in contrast with earlier releases, is a menu-driven program that takes you through a complete configuration of your system. If you run DOS 4.0 "out of the box," in normal circumstances, it takes complete control of your computer. The exact procedure you follow depends in a few details on the type of diskette drives you have. If you have 5¼-inch diskette drives, you begin by inserting the Install diskette in drive A and rebooting. This loads DOS 4.0 into your computer. You will soon see the prompts

```
Insert SELECT diskette in drive A:

Press Enter to continue installing DOS,
or press Esc to exit.
```

If you want to install DOS manually, or to explore the files on the various diskettes, press the Escape key, and you will be at an A> prompt. To continue, follow the directions. The Select program then looks for a hidden, zero-length file called DOS04L.400 on the Select diskette. If it doesn't find this file, it will abort. If it does find the file, it will continue, and shortly will ask you to insert the Install diskette again. This time it looks for a hidden, zero-length file called DOS00I.400, and if it doesn't find this file, you'll have to reboot. If it does find the file, you will soon see the screen shown in Figure 81. As you can see, if you use 5¼-inch diskettes, you *must* use 360K diskettes for the installation, even in a 1.2-megabyte drive. This means that, unless you have formatted the diskettes in advance, you will probably not be able to read them in a normal 360K drive, or, indeed, in any drive other than the one in which you format them.

If you choose to install DOS on diskettes, Select will prompt you through the making of a complete set of working diskettes. The number

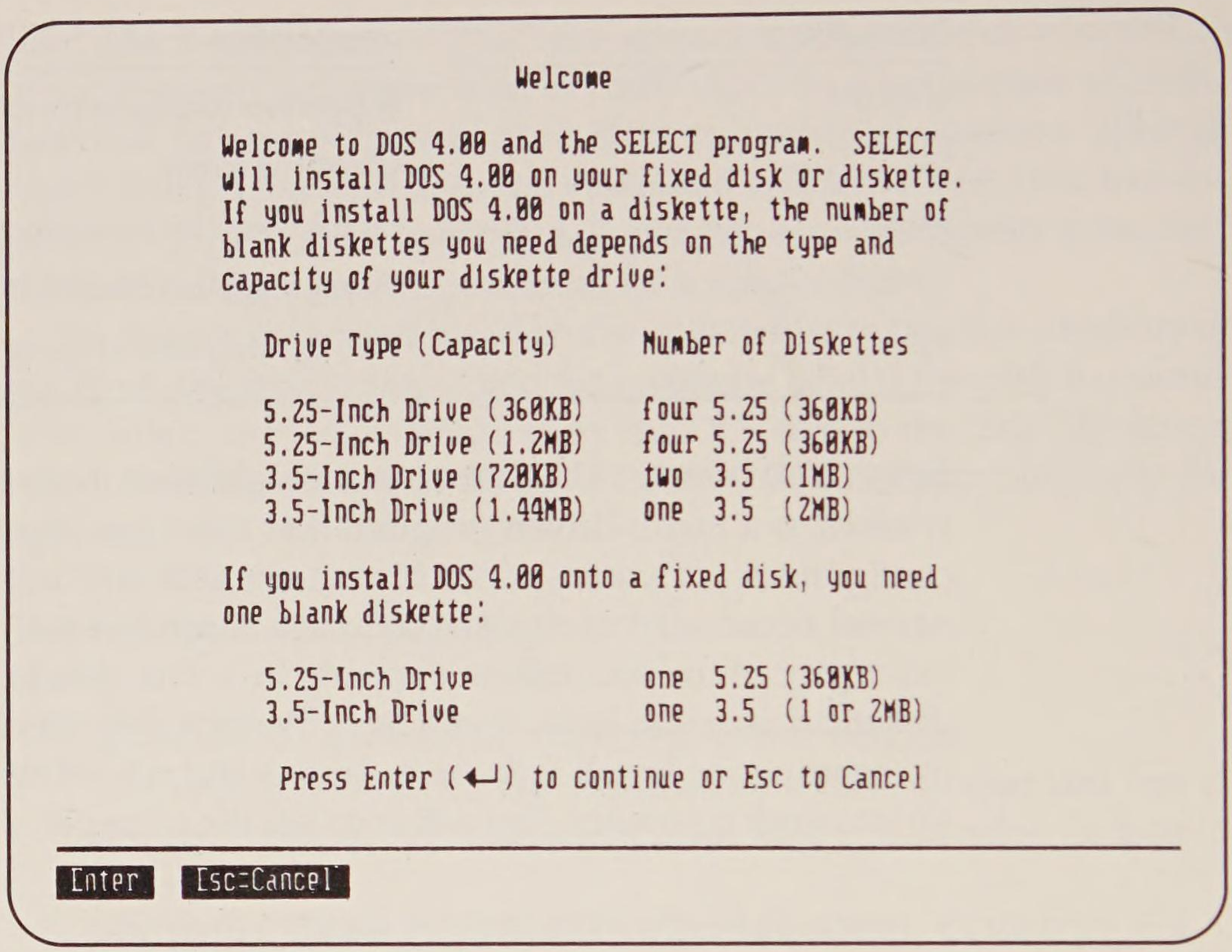


Figure 81: The Select opening screen

will depend on your drive's capacity. It will also take you through a series of choices, your responses to which it will use to build your AUTOEXEC.BAT and CONFIG.SYS files.

If you plan to install DOS 4.0 on a hard disk that isn't empty, *make a complete backup of the disk before you begin. If it has multiple partitions, back them all up*. If you install DOS 4.0 on a hard disk, you will be taken through partitioning and formatting the hard disk if necessary (see FDisk and Format). DOS 4.0 doesn't recognize multiple partitions created by anything other than itself, and I have yet to find a hard disk format that it finds acceptable, so you will probably have to format your hard disk whether you want to or not, and partition it if it contains more than one partition.

As Figure 81 shows, if you want to install DOS on a hard disk, you still need one extra diskette. Select makes a diskette called Select Copy, which is an exact copy of the Select diskette, for reasons that are not at all clear.

At the next screen, Select asks you to choose one of the following three options:

```
1. Minimum DOS function; maximum program workspace
2. Balance DOS function with program workspace
3. Maximum DOS function; minimum program workspace
```

It determines the contents of your AUTOEXEC.BAT, CONFIG.SYS, and DOSSHELL.BAT files based on your answer to this question, the hardware you say is installed, and the hardware it can detect in the system (see AUTOEXEC.BAT File, CONFIG.SYS File, and DOSShell). Note that it cannot detect expanded memory unless you have one of three particular IBM memory boards that it happens to approve of, and will not let you install it.

The default selection, 2, is highlighted. You can move the selection bar with the cursor keys. At virtually every screen, you press Return to accept a choice, Escape to go back to the previous screen, or F1 to display a help screen. The help screens are generally informative. Next, it asks whether you want to install the DOS files in the root directory, a directory called DOS400, or some other directory. DOS400 is the default. Based on your answers to these two questions, it constructs the files. (If you manage to get it to recognize your hard disk as a valid medium, and you already have CONFIG.SYS and AUTOEXEC.BAT files, it will name the new files CONFIG.400 and AUTOEXEC.400, suggesting you copy these files over your old ones.

If you accept the default values, here are the contents of CONFIG.400:

```
break=on
buffers=20
files=8
lastdrive=e
shell=c:\dos400\command.com /p /e:256
device=c:\dos400\ansi.sys
install=c:\dos400\fastopen.exe c:=(50,25)
```

The number of buffers will depend on the size of your hard disk, but everything else is fixed. Here are the contents of AUTOEXEC.400:

```
@echo off
set comspec=c:\dos400\command.com
verify off
```

```
path c:\dos400
append /e
append c:\dos400
prompt $p$g
ver
graphics
print /d:lpt1
dosshell
```

Select will create these files entirely in uppercase characters, although the style of this book demands that we show them to you in lowercase. See the various commands for explanations. Select will install the Graphics command if you tell it you have a printer, whether or not your printer can respond to the DOS Graphics command. (Only those that are IBM- or Epson-compatible can do so.)

If you choose minimum DOS function and maximum workspace, Select will omit ANSI.SYS, FastOpen, Append, Graphics, and Print, installing only the DOSShell program in memory. If you choose minimum workspace, it adds the command `fcbs=25,8` to your CONFIG.SYS file, and changes the Buffers = command to read

```
buffers=25,8
```

Having made its decisions, Select then presents you with this choice:

```
1. Accept configuration and continue with installation
2. Review, change, or add installation choices
```

The default is to continue. If you choose 2, your selections are displayed on several screens, where you can alter them. The first set of options appears in Figure 82. The second concerns the items included in the DOSShell program as shown in the DOSShell entry. The third appears in Figure 83.

When you have finally made all your choices, Select will then attempt to install itself on your hard disk. In all probability, it won't be able to, and you will see the screen shown in Figure 84, or something like it. If you choose to leave the disk unchanged, installation will probably fail. If you didn't back up all the partitions of your hard disk before you started, keep pressing Escape until you see the A> prompt. Then reboot with your old version of DOS, back up the disk, and start over (see Hard Disks, Backing Up and Restoring).

To install DOS 4.0, bypassing the Select command, and to deal with other special circumstances, see Tips.

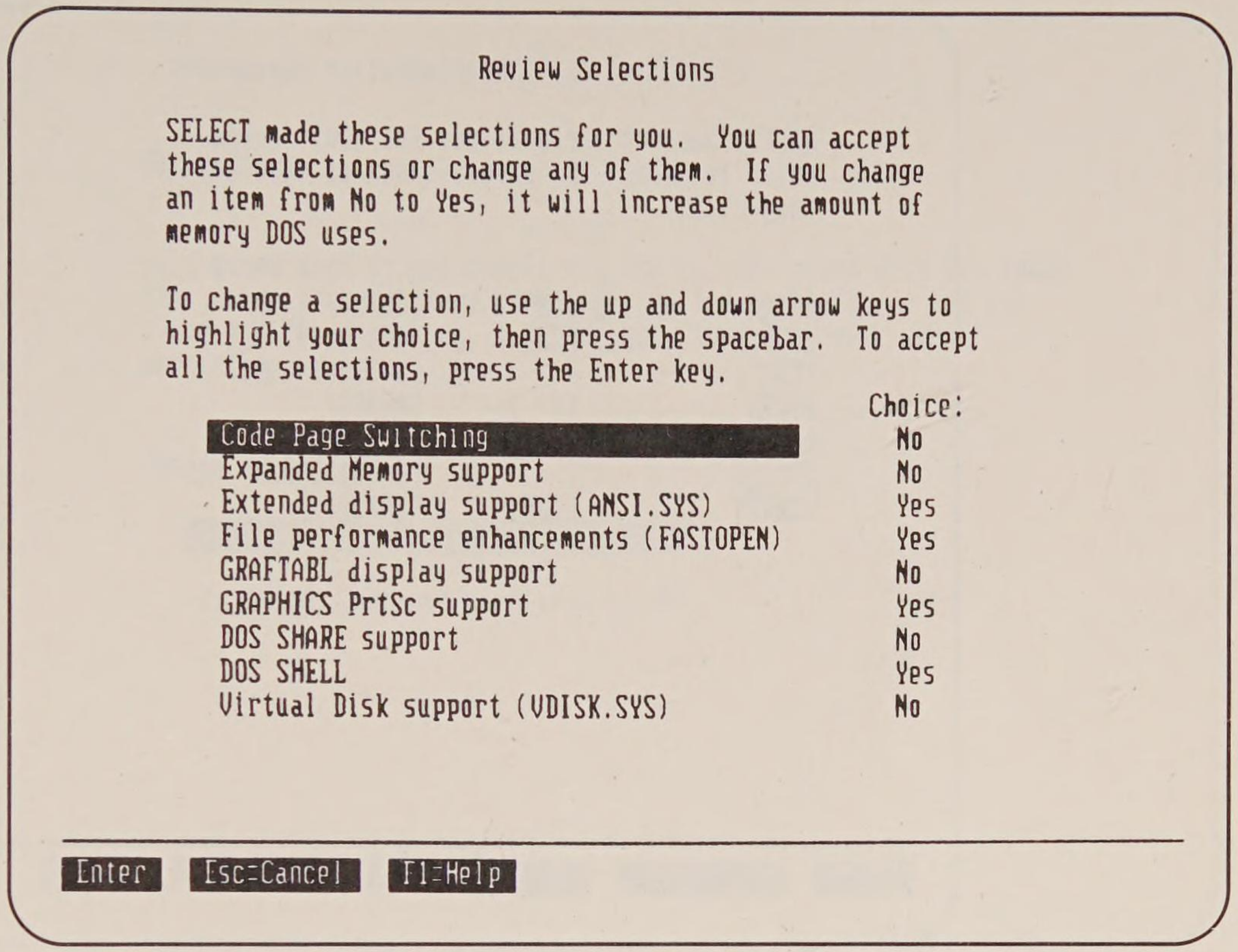


Figure 82: Select's Review Selections screen

Warnings

The Select command always erases everything on the target before doing anything else because it formats the target. Before proceeding, be sure that your target contains no files that you want to keep.

The DOS 4.0 version of Select uses the files AUTOEXEC.BAT, SELECT.COM, SELECT.HLP and SELECT.PRT on the Install diskette, and the files SELECT.DAT and SELECT.EXE on the Select diskette. You cannot simplify installing DOS by copying all of these files onto a single diskette, because it looks for two different hidden files, one on each diskette. It doesn't help to copy both files to the diskette containing all the other select files, either. If it finds the file indicating the Select diskette when it wants the Install diskette, it will lock up until you insert a diskette that contains DOS00I.400 but not DOS04L.400.

Tips

If you already have a CONFIG.SYS and AUTOEXEC.BAT file, the easiest way to nationalize your computer is to add the Country= command to your CONFIG.SYS file and the Keyb*xx* or Keyb command to your AUTOEXEC.BAT file.

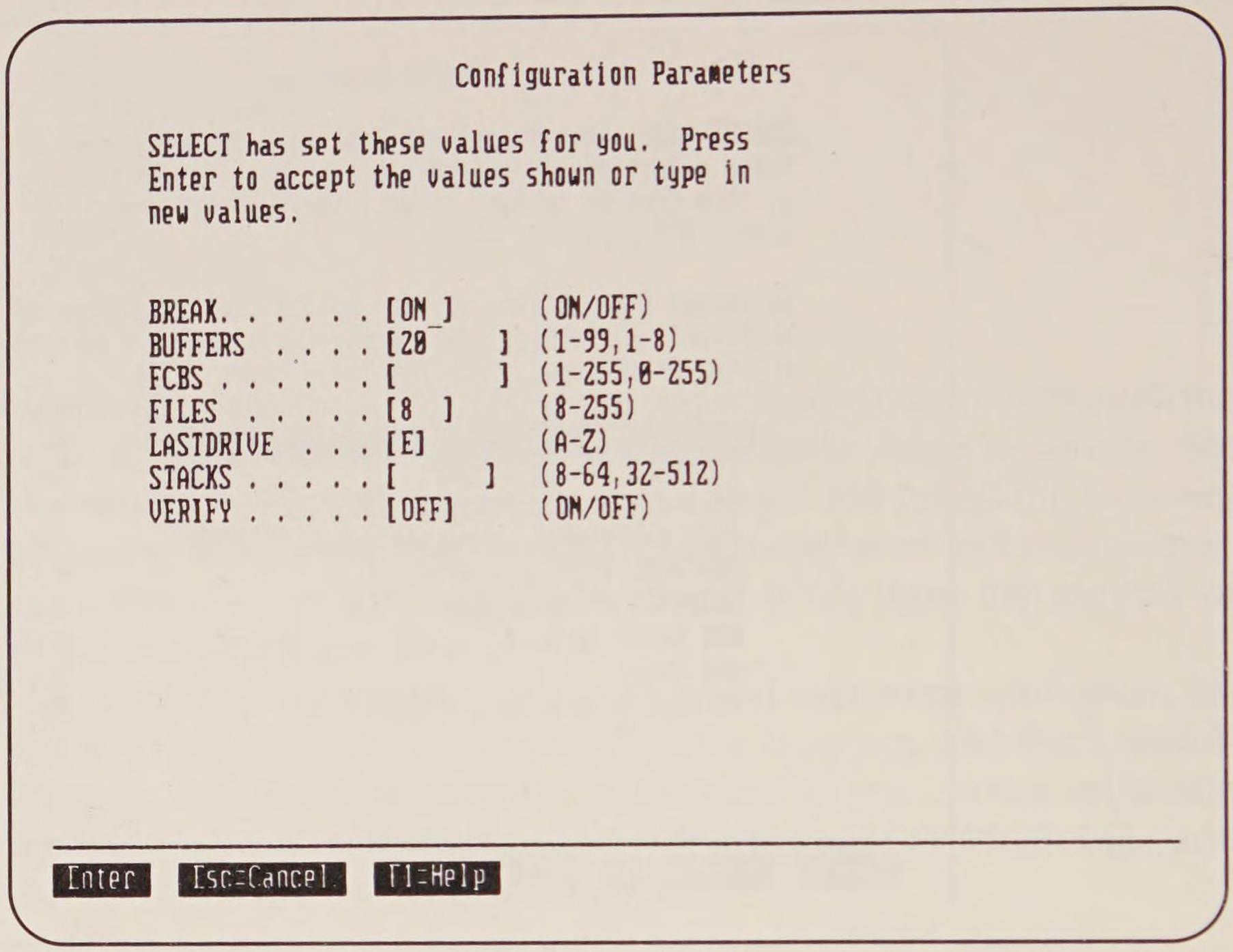


Figure 83: Select's Configuration Parameters screen

To complete the nationalizing of your computer in DOS 3.3 or later, you will probably also want to add some additional commands to your AUTOEXEC.BAT and CONFIG.SYS files. If you use an EGA or VGA adapter, or if your computer has an LCD display, you will need the DISPLAY.SYS driver (see DISPLAY.SYS). With a color graphics adapter, place the GrafTabl command in your AUTOEXEC.BAT file (see GrafTabl). Load the PRINTER.SYS driver if you use one of the IBM printers that can display any of the alternative fonts (see PRINTER.SYS). If you load either of these drivers, load the ANSI.SYS driver first.

To switch between the national default you have selected and the United States default, you must use the NLSFunc command, and load additional fonts. See NLSFunc for alternative AUTOEXEC.BAT code to complete the process of nationalizing your computer.

Installing DOS 4.0 without Using Select If you want to install DOS 4.0 on 1.2-megabyte diskettes, you must bypass the Select program. You may want to install DOS 4.0 without using the Select program for other

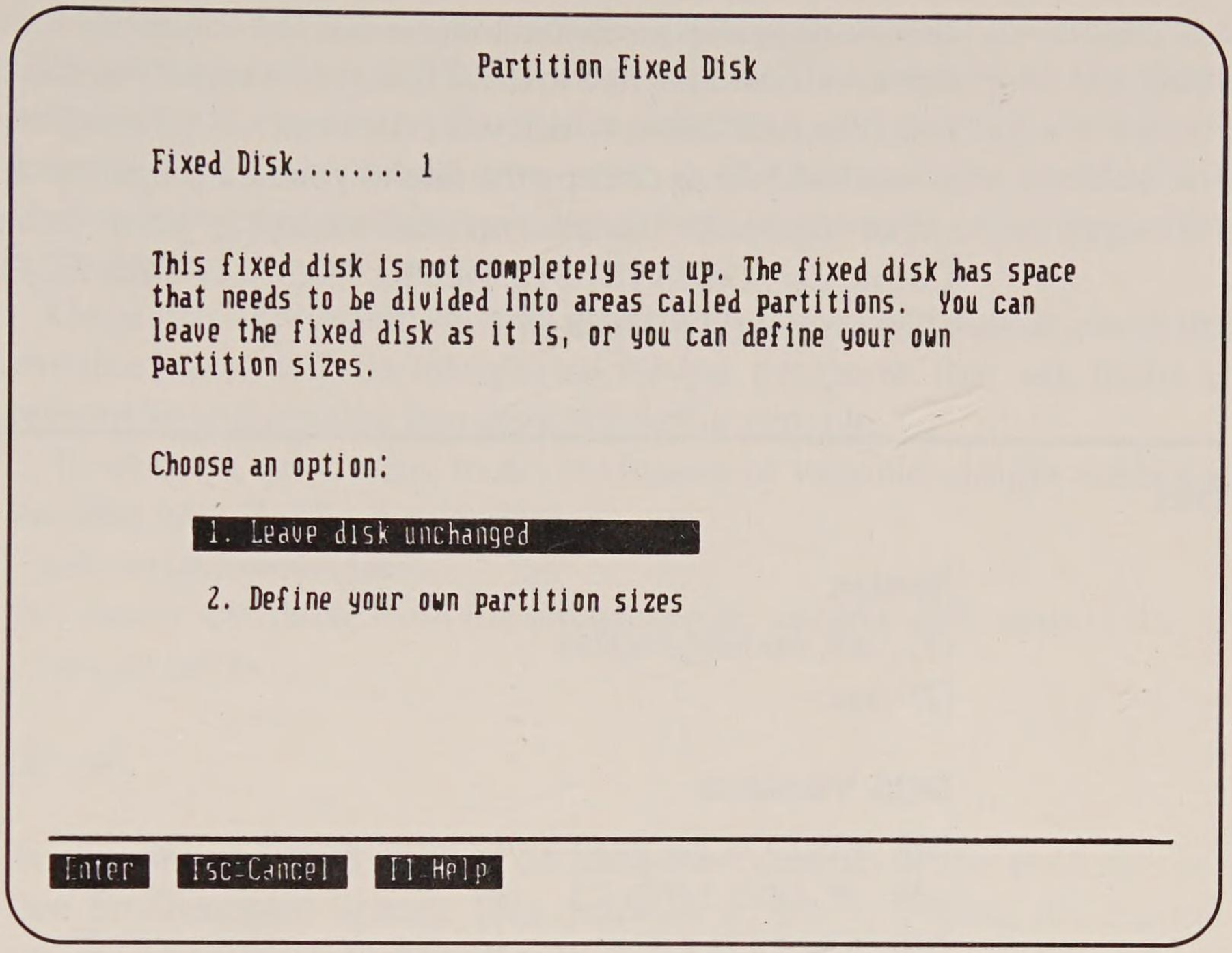


Figure 84: Select's Partition option

reasons as well—for example, if you have two diskette drives of different types, and want to make usable DOS diskettes that can be read by your B drive. These two examples require two different approaches.

If you have only one drive, and want to use 1.2-megabyte diskettes, follow these steps. Boot your system with the Install diskette, and press Escape when you see the first prompt. Enter the command `format b:/s` and replace the Install diskette with your 1.2-megabyte diskette when told to do so. You will then have a bootable high-capacity diskette. Use your regular copy procedures to copy whatever files you wish from the five DOS diskettes to your new diskette. You can fit all the DOS files onto two such diskettes. However, you cannot use the Select program unless you use a utility program that can copy zero-length files to copy DOS04L.400 to the diskette containing the Select program files.

In the second circumstance, the procedure is more complex. Suppose, for example, you have a 1.2-megabyte drive designated A and a 720K drive designated B, and you have DOS 4.0 on 3½-inch diskettes. The solution is to open your computer, find the wide, flat cable attached to your diskette

drives, and swap its connectors, so that the connector formerly attached to drive A is now attached to drive B, and vice versa. You can then boot DOS 4.0 from the 720K drive, which will now be drive A. From there, you can install it on a hard drive, or copy the files to your 1.2-megabyte drive.

For complete details on nationalizing your computer, see also Country=, COUNTRY.SYS, DISPLAY.SYS, GrafTabl, Keyb, Keyb*xx*, Mode, NLSFunc, and PRINTER.SYS.

Set

Syntax

{**1**} set *variable=value*
{**2**} set

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 1.0 to 3.3
{**all**} IBM DOS 4.0

Overview

The Set command allows you to place variables in the Environment space (see Environment Space). Once the variables are there, any program that can access the environment space can read the values assigned to the variables.

Special syntax allows you to read the values assigned to environment variables in batch files. For details on this syntax, see If.

Procedure

{**1**} set *variable=value*

In an environment variable, both the variable name and the value assigned to it are strings (sequences of characters). An environment variable ends with an ASCII 0, or Null.

The environment is specific to the command processor, or to a program currently in memory. This means that when you load a secondary command processor, it "inherits" any values that are already in the environment (see Command). However, any variables added to the environment when the secondary command processor is in effect will disappear when you exit the secondary command processor.

You may enter environment variables in uppercase or lowercase, at a DOS prompt or in a batch file. However, when referring to the value assigned to the variable, you must match the case of the characters in the value. (Regardless of how you enter it, the name of the variable will always be in uppercase, but you can refer to the name in either uppercase or lowercase.)

Once you have established an environment variable, references to the variable name will be interpreted by the programs that use them as references to the value you assigned to the variable.

To assign a new value to an environment variable, simply enter the variable name with a new value:

```
set variable=newvalue
```

To remove a variable from the environment, assign a null value to it:

```
set variable=
```

{2} set

Typing set at a DOS prompt displays the contents of the environment (see Environment Space). This includes, at the very least, the current command processor and a null path. If you have established a prompt, it too will appear in the environment space. If you used the DOS 3.3 Append command with the /e switch, your append list will appear (see Append).

Examples

Making Information Available to a Program Environment variables are used by many programs to store the location of various files. For example, you can use the Set command to tell many compilers where your libraries are stored. The command takes a form like this:

```
set lib=c:\compiler\lib
```

Testing Environment Variables in Batch Files In batch files, you refer to the value of an environment variable by placing the variable name between a pair of percent signs:

```
%variable%
```

You can use this form to test for certain values in If statements if you want to decide whether a given procedure should be carried out.

Let's say you use the windowed operating environment DESQview, and you load some applications via batch files that also load some TSRs and their files before opening the program (see TSRs). However, since DESQview

includes its own macro processor, you might want to avoid loading another macro processor, such as SuperKey. You could take the following steps. First, in your batch file that starts DESQview, include the command

```
set dv=true
```

Then, in your batch files that load programs, modify the commands that address SuperKey by testing for the environment variable DV:

```
if not %dv%==%true key filename /m1
```

This will test to see whether DESQview is present by testing the environment variable you establish when you load it. If DESQview is loaded DV is *true*. If DV is *not true*—the condition this statement tests for—SuperKey is loaded. If it is *true*, the rest of the command will not be executed, and SuperKey will not be loaded. For further details on this syntax, see If.

Viewing the Environment If you boot from a diskette and do not set a search path or modify the prompt, typing set would result in the display

```
PATH=
COMSPEC=A:\COMMAND.COM
```

If you boot from a hard disk, set up a search path and modify the prompt, and use the /e switch to the Append command (see Append), typing set would display something like this:

```
PATH=C:\;C:\DOS;C:\BATCH;C:\UTILITY
COMSPEC=C:\COMMAND.COM
PROMPT=$p$g
APPEND=C:\DICT
```

If you have established any environment variables, you will see their current values as well.

Warnings In DOS 3.0 and later, if a batch file tests an environment variable that has not yet been established, you will get a `Syntax error` message. To avoid this message, assign a dummy variable to both sides of the equation that tests the variable:

```
if %dv%@==true@ . . .
```

See If for further explanation of this technique.

Tips

Be careful how you treat spaces. In many DOS commands, it doesn't matter whether you enter spaces before or after operators. However, when you include spaces in Set commands, they are treated as part of the string in which they appear. Thus, for example, if you enter the two commands

```
set ace=one
set ace = one two three
```

and then enter a Set command, you will see

```
ACE=one
ACE = one two three
```

In the first instance, the variable name is the three-character string ACE. In the second, the variable name is the four-character string ACE, which includes a space as its fourth character. Similarly, the value assigned to the second variable is " one two three", beginning with a space.

For related information, see Append, Environment Space, ErrorLevel, If, and Path.

Setup

DOS Versions

MS-DOS 3.0 to 4.0
PC-DOS 3.0 to 3.3
IBM DOS 4.0

Overview

Setup is a program that records information about the hardware installed in computers based on the 80206 and 80386 microprocessors. It records the following information:

- The amount of conventional memory installed
- The amount of extended memory installed
- The type(s) of display adapters installed
- Which of the display adapters to use on startup, if there is more than one

- The number and kind of diskette drives
- The number and kind of hard disks
- The current date
- The current time

It stores this information on a special type of memory chip called a CMOS chip, which receives continuous power from a battery inside your computer.

You should run the Setup program any time you change your hardware configuration or, if you use DOS 3.2 or earlier, when you change the date or time.

Procedure

Different computer manufacturers handle the Setup program differently. IBM places the Setup program on the PC AT Diagnostics Disk. To run it, you must boot your computer with that disk and press the F1 key.

Some manufacturers place the Setup program on the DOS system diskette. If your DOS includes this program, run it by booting from the system diskette and typing `setup`.

Other manufacturers build the Setup program into ROM. On these computers, you run it by pressing an odd key combination such as Ctrl-Alt-Esc. Since this immediately cancels whatever you are doing and takes you to the Setup program, be sure you are not running an application when you press the keys; otherwise, you will lose your work.

All versions of Setup present you with menus and prompts to take you through the procedures step by step. Follow the prompts that your version of the program gives you.

Warnings

Some versions of Setup ask you to enter the type of hard disk you have as a number from 0 to 15. If you are the least bit unsure of the correct number for your hard disk, call your dealer before running Setup. Entering the wrong number can prevent you from accessing your hard disk, and may also cause more serious problems with your computer.

If your batteries get weak, the information on the CMOS configuration memory chip and the real-time clock chip will disappear, and your computer will tell you to run the Setup program again. When this occurs, turn off and unplug the computer. Replace the battery, then run Setup again.

Tips

Most releases of DOS 3.3 and later will change the date or time recorded in the CMOS chip when you change these items with the Date or Time commands. With these versions of DOS you do not also have to run the

Setup program to change the date or time. If the documentation doesn't say, the only way to be sure if your version writes to the CMOS chip is to issue the Date or Time command, reboot the computer, and see whether the date or time you entered is still current.

Share

Syntax

{1} share [/f:*size*][/l:*locks*]
{2} install=share.exe [/f:*size*][/l:*locks*]

DOS Versions

{1} MS-DOS 3.1 to 3.31
{1} PC-DOS 3.1 to 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

If your computer is on a network and is not the main computer on that network, you can use the Share command to control others users' access to your files. If you have created logical drives larger than 32 megabytes using the DOS 4.0 version of FDisk, you must use the Share command if DOS is to be able to read and write to your drives (see FDisk).

Once you have loaded Share, any of the other computers on the network may use the disks or directories to which you have granted access on the network. Thus, several computers may be using one of your files at the same time. If two computers each require access to a record the other is using, each will wait forever for the other to release the record. This unbreakable lockup is known as "deadly embrace."

The Share command establishes a system of locks to prevent access to records currently in use. This does not prevent deadly embrace, but at least it ensures that two computers won't both try to update the same record at the same time. The Share command also limits the number of files that can be shared at one time and the number of users that can share a file. When it is loaded, Share ensures that all attempts by other computers to read your files, or to write files on your computer, conform to the limits you have established.

Share is a TSR that takes about 5K of your RAM when it is loaded (see TSRs). It stays in memory until you reboot the computer.

Procedure

Share is an external command; using it requires having the DOS program SHARE.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\share
```

To install file sharing, or to ensure that your extra-large disk volume will be readable, enter the Share command:

```
share [/f:size][/l:locks]
```

Alternatively, if you use DOS 4.0, you can install the Share command in very low memory by loading it using an Install= command in your CONFIG.SYS file:

```
install=share.exe [/f:size][/l:locks]
```

See Install= and CONFIG.SYS File for further details.

The /f switch controls the number of files that can be accessed, and the /l switch the maximum number of locks per file. The number of locks determines the number of users who can access a single file.

If you do not enter an /f switch, the default for /f is 2048 bytes. This space is used for a table of file names being used by others on the network. For each file, the table includes the complete file specification: the drive name, full path name, and file name. It also allocates an additional 11 bytes for each file for its own purposes. If your complete file specifications are the maximum allowable size of sixty-three characters, the table can hold twenty-seven file names. On the average, a full file specification is twenty characters. Thus, the default allows for sixty-six average-length file specifications.

The default number of locks is 20. You can change either of these values by specifying the parameters on the command line; for example,

```
share /f:3064 /l:10
```

This increases the number of files that can be shared, but reduces the number of users that can share each one.

The Share command uses file control blocks to keep track of open files (see FCBs=). If you do not have an FCBs= command in your CONFIG.SYS file, Share establishes defaults of a maximum of sixteen files that can be opened using file control blocks, and eight that cannot be closed (see

CONFIG.SYS File). If you do have an FCBs= command, Share uses the number of file control blocks established by it.

Warnings

If you try to access a file currently in use on the network, you will see the message

```
Sharing violation
Abort, Retry, Ignore?
```

Press R for Retry. You should be able to gain access to the file when the current user releases it. If you press A for Abort, you will be returned to the DOS prompt. Any work in memory that you have not saved will be lost.

If the file is locked, you will see the message

```
Lock violation
Abort, Retry, Ignore?
```

Follow the same procedure. For further information, see the Abort, Retry, Ignore, Fail? entry.

Shell=

Syntax

{1} shell=[*drive*][*path*\]*filename.ext*
{2} shell=[*drive*][*path*\]command.com [/p] [/e:*nn*]
{3} shell=[*drive*][*path*\]command.com [/p] [/e:*nnnnn*]
{4} shell=[*drive*][*path*\]command.com [/p] [/e:*nnnnn*] [/msg]

DOS Versions

{1}	MS-DOS 2.1 to 3.0
{1}{2}	MS-DOS 3.1
{1}{2}	PC-DOS 3.1
{1}{3}	MS-DOS 3.2 to 3.31
{1}{3}	PC-DOS 3.2 to 3.3
{1}{3}{4}	MS-DOS 4.0
{1}{3}{4}	IBM DOS 4.0

Overview

The Shell= command is a configuration command used in the CONFIG.SYS file. It tells DOS the name of the program that will be the primary command processor (see Command and CONFIG.SYS File). This is the program that takes care of all your system's primary input and output operations. In DOS releases prior to 3.1, the command existed primarily for programmers who wanted to write an alternative to COMMAND.COM. In releases 3.1 and later, it can be used to select certain options for COMMAND.COM upon starting up your computer. You can set the same options using the Command command, but the Shell= command uses less memory, because it merely configures the original copy of COMMAND.COM loaded into the computer, rather than adding a second copy.

Procedure

The Shell= command is entered in the CONFIG.SYS file, flush with the left margin, on a line of its own; for example,

```
shell=c:\dos\command.com
```

If the file being loaded is not in the root directory of the boot disk, you must specify the complete path to the file. It's generally a good idea to specify the complete path to the file in any case.

Loading an Alternative Command Processor

{1} `shell=[drive][path\]filename.ext`

To load a command processor other than COMMAND.COM, include a Shell= statement in your CONFIG.SYS file, specifying the name and location of the file to be used as the system's primary command processor.

Configuring COMMAND.COM in DOS 3.1

{2} `shell=[drive][path\]command.com [/p] [/e:nn]`

The main reason to use a Shell= command in DOS 3.1 and later is to enlarge the DOS environment space (see Environment Space). The environment normally holds the prompt, the path, and the location of the primary command processor. However, some programs require you to place additional information in the environment space with the Set command (see Set). The environment is finite in size. If you try to place more in it than it will hold, you will see the message `Out of environment space`

and you will not be able to place anything in it. You may encounter this problem if you use a long path, a complex prompt, or a programming language or other program that requires you to use many Set commands.

To change the size of the environment, use the /e:*nn* switch, where *nn* multiplied by 16 results in the size of your intended environment in bytes. You must include the /p (for permanent) switch in the command or the AUTOEXEC.BAT file won't be executed.

Configuring COMMAND.COM in DOS 3.2 and Later

{3} `shell=[drive][path\]command.com [/p] [/e:nnnnn]`

The procedure for changing the size of the environment in DOS 3.2 and later is similar to that for DOS 3.1. However, you specify the size of the environment in bytes. The number must be a multiple of 16 between 160 and 32,768. You may need a larger environment with DOS 3.3 or later if you use the /e switch for the Append command, which places the list of appended directories in the environment (see Append). I have found that an environment of 960 to 1088 bytes is large enough to cover most contingencies, even with a long path, prompt, and append list.

Special Features in DOS 4.0

{4} `shell=[drive][path\]command.com [/p] [/e:nnnnn] [/msg]`

The DOS 4.0 Select command automatically places a Shell= statement in your CONFIG.SYS file, and selects a size for the environment based on the information you give it regarding the Path, Prompt, and Append statements you expect to use. You can, of course, modify the selected figure once you have finished installing DOS.

An additional parameter, /msg, is available in DOS 4.0. To use it, you must also use the /p switch. The /msg parameter loads all of the DOS extended error messages into memory. This parameter is useful if you use a diskette-based system, because it allows DOS to display the full text of error messages, even when the disk containing COMMAND.COM is not in the computer. Without the /msg switch, DOS will display only error numbers. You must look up the text in your DOS manual. Select automatically installs the /msg parameter if you install DOS on diskettes instead of a hard disk.

Warnings The /e switch is documented in PC-DOS 3.2 and 3.3, but not in the equivalent MS-DOS versions, nor in any version of DOS 3.1. However, I've had no trouble using it with any of these releases.

Tips If you use several versions of DOS at different times, you can use the Shell= command to simplify their relationship with a hard disk. You can keep all the DOS files (except the hidden system files) that you use for each version in separate directories—say, \DOS310 and \DOS330. Include COMMAND.COM in each directory and use the Shell= command in the CONFIG.SYS file on each boot diskette to direct your computer to the correct directory. Thus, on your DOS 3.1 system diskette, you would include the command

```
shell=c:\dos310\command.com
```

On your DOS 3.3 diskette, you would replace this command with

```
shell=c:\dos330\command.com
```

This makes it unnecessary to keep your system diskette in drive A while you work.

To use this technique with DOS 4.0, be sure that the hard disk is formatted with the earlier release you plan to use and that the OEM ID in the boot record matches the one used by your version of DOS 4.0. With any other arrangement, one of your versions of DOS will not be able to read the hard disk. (This will not work at all if your other version of DOS is Compaq DOS 3.31.)

Shift

Syntax

```
shift
goto label
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview Shift is a batch file command used to execute series of parameters. To use the command, enter the name of the batch file followed by a series of parameters. The batch file acts on the first parameter until it reaches a

Shift command. It then discards the first parameter on the command line, making the next parameter the first one. In effect, it shifts the parameters one place to the left.

Shift allows you to pass more than nine parameters to a batch file. The command is generally used within a loop bounded by a GoTo statement and a label (see GoTo). The loop processes each successive command-line parameter in turn.

Procedure

The general form of a loop that uses the Shift command is

```
:label
command1 %1
.
.
.
commandn %1
shift
goto label
```

The boundaries of the loop are *:label* at the beginning and the GoTo statement at the end. Any number of commands may appear within the loop. The replaceable parameter %1 may also appear any number of times (see Batch Files).

To keep the loop from repeating forever, you should also include a test to see whether there are any additional parameters. This test takes the following form:

```
if "%1"=="" goto label2
```

This command tests whether the current parameter is a *null string*—a string with no characters, which is to say no parameter (see If). If there are no more parameters, execution is directed out of the loop so that the program can continue. The general form a loop with this test is

```
:label1
if "%1"=="" goto label2
command1 %1
.
.
.
commandn %1
```

```
shift
goto label1
:label2
```

Examples

The program MCOPY.BAT, which follows, copies a group of files to drive A. The advantage of this program is that it can copy a series of files that cannot be specified with a wildcard pattern (see Wildcard Characters). To invoke the program, type

```
mcopy file1.ext file2.ext file3.ext. . .filen.ext
```

Since you can substitute any single syntax unit for a replaceable parameter, you can include drive names, path names, and wildcard characters in the file names; for example,

```
mcopy b:ascii.txt c:\reports\*.wk1 c:\reports\*.wp
```

MCOPY.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`:start`	Beginning of loop
`if "%1"=="" goto end`	Tests for absence of a parameter; if no more parameters are on the command line, exits the loop
`copy %1 a:`	Copies next file named on command line
`shift`	Discards first parameter
`goto start`	Repeats the loop
`:end`	Label to find on exit from loop

When you execute this program, the first file name on the command line is passed to the %1 replaceable parameter. If the file name includes wildcard characters, the program will copy every matching file before execution proceeds to the Shift command. At that point, the first file name is discarded, and the second one becomes the first. Execution is then directed to the `:start` label, and the loop repeats. It keeps repeating until the If statement finds no more parameters. Then execution continues at the `:end` label. Since there are no more commands in the file, execution ends.

You can make this program more flexible. Include the target drive and/or directory as a parameter, and use the Set command to place it in the environment (see Set). You can then read it again each time you need it. The following program is a version of MCOPY.BAT with this refinement. To execute this version of the program, your first parameter must specify the target of the copy. For example, to copy everything to drive A as before, enter

```
mcopy a: b:ascii.txt c:\reports\*.wk1 c:\reports\*.wp
```

MCOPY.BAT, Version 2, with an Environment Variable

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`set target=%1`	Places the first parameter in the environment
`shift`	Discards the name of the target
`:start`	Beginning of loop
`if "%1"=="" goto end`	Tests for absence of a parameter; if no more parameters are on the command line, exits the loop
`copy %1 %target%`	Copies next file named on command line to the specified drive\directory
`shift`	Discards the parameter just used
`goto start`	Repeats the loop
`:end`	Label to find on exit from loop
`set target=`	Removes the environment variable

There is neither a default nor a way to test whether the first parameter is a legitimate target of the copy. If you forget to specify the target of the copy, whatever you entered as the first parameter will be the target of the copy. If it's a file name, each file specified by the other parameters will be copied to that file, replacing the previous one.

Warnings If you do not include a test for the absence of a parameter in your loop, the loop will repeat until you press Ctrl-Break.

Sort

Syntax

{1} sort [/r][/+c]<*filename.ext*
{2} *command*¦sort [/r][/+c]

DOS Versions

{**all**} MS-DOS 2.1 to 4.0
{**all**} PC-DOS 2.0 to 3.3
{**all**} IBM DOS 4.0

Overview

Sort is a filter program (see Filters) used to sort lines of text in text files. It can sort in ascending or descending order, and can sort on any column you specify. It normally directs its output to the screen, but you can redirect it to a file, a printer, or another program (see Redirection and Pipes).

In DOS 2, Sort uses the ASCII values of the characters in the chosen column as the basis on which to sort (see ASCII Code, and the tables in Appendix B). Thus, all uppercase characters appear before all lowercase characters. (Uppercase *Z* has an ASCII code of 90; lowercase *A* has an ASCII code of 97.)

In DOS 3.0 and later, the sorting order used by Sort is not the strict ASCII order. Rather, it places uppercase and lowercase versions of the same character in sequence. In addition, it places accented characters with high-order codes in the proper sequence. Accented *A*s will appear among the *A*s, accented *E*s among the *E*s, and so on.

Let's consider an example. Suppose you have a file containing the lines

Many
More
dogs
don't
eat
Big
Boys

A true ASCII sort, as in DOS 2, would produce

Big
Boys
Many
More
dogs
don't
eat

The DOS 3.0 and later Sort command, on the other hand, would produce the order

Big
Boys
dogs
don't
eat
Many
More

Numbers are sorted according to their ASCII codes in all versions of DOS. Thus, 0 comes before 1. If you sort a list containing the numbers

21
9
2116
32
90

The result would be

21
2216
32
9
90

On the other hand, the space character has a lower ASCII code than any of the numbers. Therefore, if your numbers are right-justified with spaces to the left of the first digit, the order would be correct. The list

```
  21
2216
  32
   9
  90
```

would be sorted correctly.

Procedure

Sort is an external command; using it requires having the DOS program SORT.EXE) on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
dir b:¦a:sort
```

If use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\sort<textfile.txt
```

{1} `sort [/r][/+c]<`*`filename.ext`*

To sort the text in a text file, use the file as redirected input to the sort command. To sort in reverse order, use the /r switch. To sort on a column other than the first, use the */+c* switch, where *c* is the number of the column to sort on. The column at the left margin is 1, not 0.

{2} *command*`¦sort [/r][/+c]`

You can sort the output of any command that sends output to the screen. For example, to display the current directory in sorted order, enter the command

```
dir¦sort
```

The same switches work with either syntax.

Examples

To sort the current directory by extension rather than by name, sort on the tenth column:

```
dir¦sort /+10
```

The result is a directory sorted by extension. However, if it is a subdirectory, the two entries for the root and the current directory will not be in their usual position at the top of the directory, nor will the other messages be in their usual positions. (Since you sorted on the tenth column, the character in the tenth column of the messages is sorted alphabetically among the extensions.) Figure 85 shows part of a directory sorted on the tenth column, with the tenth column highlighted.

The Sort command is most useful for sorting text in columns or fixed fields. For example, if you have a phone list in which you have been entering numbers and names at random but in regular columns, you can create a version of the list sorted by name by using Sort on the column where the names appear. Figure 86 shows such a list, a file called PHONE.DIR, before sorting. Since the names begin in the thirty-third column, use the switch /+33. The command

```
sort /+33<phone.dir
```

will display the file on the screen as shown in Figure 87. To save the sorted list to a new file, enter the command

```
type phone.dir¦sort /+33 >newphone.dir
```

The command

```
sort/+33<phone.dir >newphone.dir
```

will also work.

```
TEMP3            989    1-27-88   12:11p
TEMP2            990    1-27-88   12:00p
       32 File(s)   3102720 bytes free
SUBMIT   1      1476   10-13-87    2:44p
CULLINAN LTR    3072    9-22-87    2:16p
MCD1013  LTR    3072   10-13-87    2:04p
ADDRESS  MCD     256   10-13-87    2:15p
 Volume in drive C is JK'sTurboAT
CONVENTI ONS    1482   11-19-87    4:05p
 Directory of  C:\TEMP\TEST
```

Figure 85: A directory sorted by extension

```
Software        1(943)545-3611  Software Express
Hardware        1(472)859-6531  CBC Computer Systems
Hardware        1(472)438-8400  Computer International
Software              555-5782  Downtown Center Software
Software        1(472)649-3896  Micro Research, Inc.
Tech Support    1(472)646-6464  Micro Research Tech Support
Tech Support    1(472)411-5000  Toshiba America
Doctor                555-7600  Payne, Dr. Franklin
Office supplies       555-3576  Kerry's Office Supplies
Car repair            555-9865  Marshall's Garage
Insurance             555-2311  Borkow, Sam
Dentist               555-6232  Filler, Dr. Edward Z.
```

Figure 86: An unsorted phone list

```
Insurance             555-2311  Borkow, Sam
Hardware        1(472)859-6531  CBC Computer Systems
Hardware        1(472)438-8400  Computer International
Software              555-5782  Downtown Center Software
Dentist               555-6232  Filler, Dr. Edward Z.
Office supplies       555-3576  Kerry's Office Supplies
Car repair            555-9865  Marshall's Garage
Tech Support    1(472)646-6464  Micro Research Tech Support
Software        1(472)649-3896  Micro Research, Inc.
Doctor                555-7600  Payne, Dr. Franklin
Software        1(943)545-3611  Software Express
Tech Support    1(472)433-5999  Toshiba America
```

Figure 87: A sorted phone list

Warnings

Beware of files that are not true ASCII files. If they contain either control characters or formatting information in the form of high-order characters, the results of the Sort command will be unpredictable (see Type and ASCII code).

A peculiarity of the Sort command is that it requires spaces before the operator characters at certain points. Thus, although

```
sort/+33<phone.dir >newphone.dir
```

works as expected, if you leave out the space before the > character you get a `File not found` message.

Tips

You can take advantage of the fact that right-justified numbers are sorted correctly to sort directories by size. Use the command

```
dir|sort/+14
```

For related information, see Filters, and Redirection and Pipes.

Stacks=

Syntax

stacks=*frames*,*size*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.1 to 3.3
IBM DOS 4.0

Overview

Stack frames are used to process hardware interrupts. Whenever an interrupt is occurs (virtually every DOS process involves an interrupt at some point), a stack frame is used. When the interrupt has finished processing, the stack frame is released. Stack frames can range in size from 32 to 512 bytes each.

The Stacks= command changes the default number and size of the stack frames. On PCs, PC XTs, and compatibles, the default number of stack frames is 0. On PC ATs and 386 PCs, the default is nine 128-byte stack frames.

You should not need to change these defaults unless you encounter the message

```
FATAL ERROR: Internal stack overflow.
System Halted.
```

You are likely to encounter this error only if you have many devices installed in your system.

Procedure

Stacks= is a configuration command, entered via the CONFIG.SYS file (see CONFIG.SYS File). To change either the number of stack frames or their size, enter the command

stacks=*frames*,*size*

where *frames* is the number of frames (from 8 to 64) and *size* is the size of each frame in bytes (from 32 to 512). Each additional stack frame uses some of your RAM, leaving less available for applications.

Tips

If you need to increase the number of stacks, the most commonly suggested value is `stacks=16,128`.

Subst

Syntax

{1} subst *virtualdrive drive\path*
{2} subst *virtualdrive* /d
{3} subst

DOS Versions

{**all**} MS-DOS 3.1 to 4.0
{**all**} PC-DOS 3.1 to 3.3
{**all**} IBM DOS 4.0

Overview

The Subst command (short for substitute) assigns a new drive name to the name of an existing directory, thus creating a *virtual drive*, or *logical drive*, with a new drive name. This command is useful when you want to refer to a long path name with something shorter. It also allows you to use subdirectories with a program that can change from one disk to another but does not recognize path syntax.

Procedure

Subst is an external command; using it requires having the DOS program SUBST.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\subst e: c:\reports\south\qtr2
```

If you do not have a LastDrive= statement in your CONFIG.SYS file, the last drive name that can be used is E (see LastDrive=). If you already have drives A through E in your system, you must use the LastDrive= command if you want to create additional virtual drives. If the drive name you select is out of range, you will see the message

```
Invalid parameter
```

You will also see this message if you try to use the same drive name in a second substitution statement.

When a substitution is in effect, you can refer to the directory either by its full path name or by its new drive name. Thus, the command

```
dir e:
```

results in a display beginning with lines such as

```
Volume in drive E is MY SYSTEM
Directory of E:\
```

At the same time, the command

```
dir c:\reports\south\qtr1
```

results in a display beginning with

```
Volume in drive C is MY SYSTEM
Directory of C:\REPORTS\SOUTH\QTR1
```

The rest of the display is the same for either command.

Creating a Virtual Drive

{1} `subst` *`virtualdrive drive\pathname`*

To make a directory into a virtual drive, enter the Subst command, followed by the name to be given the virtual drive, and the drive and path name to which the new name will be given, separated by spaces. For example,

```
subst d: c:\invoices\east\qtr4
```

The drive name can be the name of an existing drive in your system. However, you should choose a name that isn't already in use so that you won't lose access to the physical drive of the same name. Note also that the names available are limited either to A through E if you have not entered a LastDrive = statement, or to A through the letter used in your LastDrive = statement.

When you have entered the preceding command, you can make the \INVOICES\EAST\QTR4 directory current by entering the command

```
d:
```

Canceling a Substitution

{2} `subst virtualdrive /d`

To cancel the assignment made with a Subst command, enter the command again, followed by the name of the virtual drive and /d (delete) switch:

```
subst d: /d
```

You can also cancel substitution assignments by rebooting the computer.

Viewing the Substitutions in Effect

{3} `subst`

Entering the Subst command with no parameters produces a report of the substitutions in effect. If you issue the Subst commands described already, deleting either substitution, this command will display

```
D: => \INVOICES\EAST\QTR4
E: => \REPORTS\SOUTH\QTR2
```

Examples

Suppose you normally use a RAM disk named D for a group of system startup files (see VDISK.SYS, AUTOEXEC.BAT File, RAMDRIVE.SYS). In some situations, however, you might need the memory that it occupies, so you don't load the RAM disk driver, but you still need access to the files you normally place on the RAM disk. If the RAM disk is part of your Path command (see Path), and the files are normally copied to the RAM disk from a directory called \STARTUP, you can keep the startup files accessible by entering the command

```
subst d: c:\startup
```

If you use WordStar 3.3 on a hard disk, you have probably encountered its obstinate refusal to recognize path names. You can overcome this limitation and also regain access to your most used utilities through WordStar's Run a Program command by issuing the following three commands:

```
subst d: c:\wordstar
subst e: c:\dos
subst f: c:\utility
```

You can then make whatever data directory you wish current, run WordStar from drive D, and run your DOS and utility programs by prefacing them with a drive name. Of course, you must also have the command

```
lastdrive=f
```

in your CONFIG.SYS file.

Warnings

Subst is a dangerous command. When a substitution is in effect the Assign, Backup, ChkDsk, DiskCopy, FDisk, Format, Join, and Restore commands work erratically, unpredictably, or not at all. Subst is also dangerous if one of the drives involved in a substitution is part of a network. At best, you may be temporarily unable to access a drive. At worst, you may accidentally overwrite files with the same name in a different directory or on another computer. Subst can also lock up your computer completely.

Tips

If you have heavily nested directories and you want to put some of them on the search path or the append list, you can shorten the path statement by using a Subst command referring to one of the first-level directories. You can then refer to subsidiary directories as if they were first-level directories. Thus, if you enter the command

```
subst d: c:\invoices
```

you can then include the following on your search path or append list:

```
d:\south;d:\east;d:\midwest;d:\west
```

See Append, Directories and Path for further details and a map of the directory to which these commands refer.

Switchar

Syntax

```
switchar=x
```

DOS Versions

MS-DOS 2.1 to 2.11

Overview

Switchar is an undocumented configuration command designed for computers that use MS-DOS but are not IBM compatible.

Some computers do not have a backslash character on the keyboard. MS-DOS releases for these computers use the slash (/) as the path separator in path names. Since this character cannot also be used to indicate command switches, the command switch character on these computers is the hyphen character (-).

The Switchar command lets you choose whether the switch character is a hyphen or a slash. If you use the slash, the backslash is the path separator. If you use the hyphen, the slash is the path separator.

Procedure

To change the switch character, enter the Switchar command in your CONFIG.SYS file (see CONFIG.SYS File), followed by one of the acceptable parameters:

```
switchar=-
```

or

```
switchar=/
```

Examples

If you use the command

```
switchar=-
```

in your CONFIG.SYS file, a command normally entered as

```
c:\dos\chkdsk /f
```

must to be entered as

```
c:/dos/chkdsk -f
```

This can be a definite convenience on some keyboards. In some non-IBM compatibles using MS-DOS, for example, you can enter a backslash only as an Alt-key combination. It can also be helpful if your backslash key is in an awkward location for touch typing.

Switches

Syntax

```
switches=/k
```

DOS Versions

MS-DOS 4.0
PC-DOS 4.0

Overview

The Switches command allows applications that have no provision to use the enhanced (101-key) keyboard to operate as though a standard keyboard were installed.

Procedure

To force an application to use an enhanced keyboard when it normally will not, enter the command

```
switches=/k
```

This will cause the numeric keypad and the cursor keypad to use the same keyboard scan codes and extended ASCII codes, and disable the F11 and F12 keys.

Sys

Syntax

{**1**} sys *drive*
{**2**} sys *source drive*

DOS Versions

{**1**} MS-DOS 2.1 to 3.31
{**1**} PC-DOS 1.0 to 3.3
{**all**} MS-DOS 4.0
{**all**} IBM DOS 4.0

Overview

The Sys command writes the operating system files to a formatted disk. It may or may not also copy COMMAND.COM to the target disk. If the disk is already a system disk, you can use the Sys command to update the operating system to a later version.

Procedure

Sys is an external command; using it requires having the DOS program SYS.COM on a disk in your computer. If you use DOS 2, it must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:sys b:
```

If you use DOS 3.0 or later and the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\sys a:
```

To transfer the operating system to a blank, formatted disk, or to update the existing operating system on a system disk, first check that the current drive contains the hidden system files, then enter the Sys command followed by the name of the target drive. This command copies the hidden system files IBMBIO.COM and IBMDOS.COM (or IO.SYS and MSDOS.SYS) to the target disk. Some versions of the Sys command also copy COMMAND.COM to the target disk.

In DOS 4.0, the hidden system files need not be on the default drive. Thus, for example, if drive B is current, you could transfer the system from drive A to drive B with the command

```
sys a: b:
```

Warnings

When you attempt to transfer the system to a disk with the Sys command, you may see the message

```
No room for system on destination disk
```

This message means one of two things: First, the target disk is not now a system disk; also, the part of the disk immediately following the root directory (where the system files must appear) is already occupied by other files. If this is true and you can determine which files follow the root directory, you can copy them and delete the originals to make room for the system. (Note that the files in question are not necessarily the first ones in the directory.)

Second, you may be unable to transfer the system because the disk was formatted with an incompatible version of DOS. The names of the hidden system files used by the DOS that formatted the disk are part of the boot record (see Boot Record). If the DOS that formatted the disk calls these files IBMBIO.COM and IBMDOS.COM and your DOS calls them IO.SYS and MSDOS.SYS (or vice versa), you will be unable to transfer the system unless you reformat the disk. Also, because the boot record created by DOS 4.0 is different from that of earlier DOS releases, you may not be able to transer the DOS 4.0 system files to a disk that was formatted with an earlier version of DOS.

Tips

You can transfer the system to a disk when you format it by using the `format /s` command. This command always copies COMMAND.COM as well as the hidden system files.

For related information, see Disk Clusters and Format.

Text, Displaying

For methods of displaying text, see ANSI.SYS; Echo; Graphics Characters, Displaying and Printing; Prompt; Rem; and Type.

Time

Syntax

{1} `time`
{2} `time` *hh*`[:`*mm*`:`*ss*`.`*cc*`]`
{3} `time` *hh*`[:`*mm*`c`

DOS Versions

{1}{2} MS-DOS 2.1 to 3.3
{1}{2} PC-DOS 1.0 to 3.31
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

The Time command displays the current time as DOS understands it and allows you to change the current time. The system computes time on a twenty-four-hour basis. It may appear otherwise, however, since directories show the time on a twelve-hour basis, with an *a* or *p* to indicate before or after noon. The time advances eighteen times a second from the time you start your computer, starting at the time known to DOS at startup. Unless your computer has a real-time clock, this is the creation time of the operating system files under MS-DOS, or 12 PM under PC-DOS or IBM DOS, until you tell your computer otherwise.

An installable real-time clock comes with either a program or a device driver that reads the time on that clock and transmits the current time to DOS. If you have a PC AT or compatible or a 386 PC, the time is stored in the CMOS memory chip. DOS automatically reads the time from this chip when you boot the computer. To change the time recorded in this chip you must use the Setup program, unless you use DOS 3.3 or later. See Tips.

It is useful for your computer to know the current time (but not as crucial as having it know the current date, since the Backup and Restore programs in DOS 3.2 and later can select files to back up or restore based on the directory date). More important, you may need to know the correct time that a file was last changed in order to find a particular iteration of the file.

Procedure

{1} `time`

If you do not use an AUTOEXEC.BAT file (see AUTOEXEC.BAT File), your computer displays a prompt of the form

```
Current time is hh:mm:ss.cc

Enter new time:
```

(*cc* represents hundredths of a second) before displaying the DOS sign-on message. You can enter the time to any degree of precision up to hundredths. Entering

```
10
```

is just as acceptable as entering

```
10:00:00.00
```

However, because DOS uses a twenty-four-hour clock, to set the time to 1 PM, you must enter

```
13
```

unless you use DOS 4.0. This release allows you to enter the time as it appears in directory entries. Thus, in the United States, you could enter 1 PM by entering

```
1:00p
```

at the Time command's prompt. If your country displays the date in some other form, use your country's time format.

If you do use an AUTOEXEC.BAT file, neither the time prompt shown above nor the DOS sign-on message appears because DOS immediately begins executing the commands in the AUTOEXEC file. Your default system time then becomes the time known to DOS.

If you have a real-time clock or a PC AT or compatible or 386 PC, this time will be correct. However, if you have neither, and you want your directory to be accurate, or you have programs that execute at a specified time, you should include the command `time` in your AUTOEXEC file, preferably near the beginning. This command displays the same time prompt as appears automatically when you do not use an AUTOEXEC file, and allows you to enter a new time.

If the time displayed in the prompt is correct, just press Return and it will remain the system time.

{**1**} `time`
{**2**} `time` *hh*[:*mm*:*ss.cc*]

To change the system time, enter the Time command at a DOS prompt, either with no parameters or with the desired time as a parameter. If you enter the Time command with no parameters, the current time appears, with a prompt to enter a new time. If you enter a time parameter, no messages appear. You need not include leading zeros in the hour.

If you enter the time in an unacceptable format, the prompt

```
Invalid time
Enter new time:
```

appears.

The time is stored in the directory in a two-byte number in bytes 24 and 25 of the directory entry. To decode this number, convert the hex values to decimal and use the following formula:

```
TIME = (seconds / 2) + (minutes x 32) + (hours x 1024)
```

Tips

If you have nationalized your computer for a country other than the United States, enter the time in the format for your selected country. This is also the format in which the time is displayed.

Follow these steps to change the directory time of a file: First, use the Time command to set the system time to the time desired; next, open and close the file with the program that created it; finally, use the Time command to reset the system time to the current time.

Some MS-DOS 3.3 versions of the Time command write the time directly to the CMOS memory chip on a PC AT or compatible computer or a 386 PC. Thus, you can use it to set the time permanently and bypass the Setup program. DOS 4.0 will do the same. To find out if your version of DOS updates the clock chip, change the time using the Time command and reboot the computer. If the current time on your computer reflects the time you just entered, you have updated the clock chip.

For related information, see Date.

Tree

Syntax

{**1**} `tree [`*drive*`][/f]`
{**2**} `tree [`*drive*`][`*path*`][/f][/a]`

DOS Versions

{**1**} MS-DOS 2.1 to 3.31
{**1**} PC-DOS 2.0 to 3.3
{**all**} MS-DOS 4.0
{**all**} IBM DOS 4.0

Overview

The Tree command provides a listing of the directories on the target disk and optionally lists the files in each directory.

Procedure

Tree is an external command; using it requires having the DOS program TREE.COM (or TREE.EXE) on a disk in your computer. If you use DOS 2, it

must be either in the current directory of one of your disks or on the path. If it is not on the current disk, you must preface it with a drive name; for example,

```
a:tree c:
```

If you use DOS 3.0 or later, and the program is neither in the current directory or on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\tree /f
```

To display a list of all the directories on a disk, enter the Tree command, followed by the name of the drive for which you want the list:

```
tree c:
```

If you want the list to include the names of the files in each directory, use the /f switch:

```
tree c:/f
```

In DOS 2, the latter command lists the files in the subdirectories, but not the files in the root directory.

In DOS 4.0, the Tree command produces a diagram of the directory tree, using graphics characters to construct the diagram. The /f switch lists the files in each directory within the diagram, and the /a switch substitutes low-order ASCII characters for the graphics characters, so that you can display the diagram using national language character sets that do not include the graphics characters used in the diagram or print it on a printer that doesn't print high-order ASCII characters. (See ASCII Code and ChCP for further details.)

DOS 4.0 lets you view a portion of your tree structure by specifying the directory with which the display should begin. Thus, for example, the command `tree \sales /f` would display the files in the \SALES directory, and in all its children, but not in any other directories.

Examples

The Tree command, without a switch, might produce a listing like the one shown in Figure 88. Figure 89 shows sample output of the `tree /f` command. Figures 90, 91, and 92 show the same directories displayed with the Tree command from DOS 4.0 with no switches, with the /f switch, and with the /a switch, respectively.

Tips

Compare Figure 89 with the output of `chkdsk /v` in the ChkDsk entry (Figure 12). Both figures represent the same portion of the same disk. As

```
C:\>tree

DIRECTORY PATH LISTING FOR VOLUME JK'sTurboAT

Path: \DBASE

Sub-directories:  FLASH


Path: \DBASE\FLASH

Sub-directories:  None


Path: \ACCOUNTS

Sub-directories:  1987


Path: \ACCOUNTS\1987

Sub-directories:  None

Path: \BATCH

Sub-directories:  None


Path: \DOS

Sub-directories:  EGA


Path: \DOS\EGA

Sub-directories:  None


Path: \BUSINESS

Sub-directories:  None


Path: \BASIC

Sub-directories:  ALLIANCE
                  COMPILER
                  GRAPHICS
```

Figure 88: Partial output of the Tree command in DOS 2 and 3

```
C:\>tree /f

DIRECTORY PATH LISTING FOR VOLUME JK'sTurboAT

Files:              COMMAND .COM
                    CONFIG  .WIN
                    CONFIG  .DV
                    CONFIG  .SYS
                    ANSI    .SYS
                    ANSI43  .SYS
                    DM      .SYS
                    DRIVER  .SYS
                    MSMOUSE .SYS
                    RAMDRIVE.SYS
                    X2CDRIVE.SYS
                    EMM     .SYS
                    X2CEMM  .SYS
                    AUTOEXEC.BAT
                    CAROUSEL.OPT
                    LASTBKUP
                    MIRROR  .FIL
                    SPOOL   .COM
                    TREEINFO.NCD

Path: \DBASE

Sub-directories:  FLASH

Files:              ASSIST  .HLP
                    DBASE   .COM
                    DBASE   .LD1
                    DBASE   .MSG
                    DBASE   .OVL
                    DBASEINL.OVL
                    HELP    .DBS
                    CONFIG  .DB
                    DBFIX   .EXE
                    DBLOG   .FIX
                    DBLOGM  .FIX
                    DBASE   .EXE

Path: \DBASE\FLASH

Sub-directories:  None

Files:              FL      .COM
                    FLASHUP .COM
                    FLCLIP  .OBJ
                    FLDEFALT.SYS
                    FLEDITOR.000
                    FLEDITOR.CHN
                    FLGEN   .DB2
                    FLGEN   .DB3
                    FLHELP
                    FLHLPWIN
                    FLINSTAL.CHN
                    FLMAIN  .CHN
                    FLMAKDB2.000
                    FLMAKDB2.CHN
                    FLMAKDB3.000
                    FLMAKDB3.CHN
                    FLMAKHLP
                    FLMAKPRO
                    FLMAKWIN.000
                    FLMAKWIN.CHN
                    FLMENUS .WIN
                    FLSCREEN

Path: \DICT

Sub-directories:  None
```

Figure 89: Partial output of the Tree /f command in DOS 3

```
C:\>tree

Directory PATH listing for Volume JK'sTurboAT
Volume Serial Number is 2C06-003C
C:.
├───DBASE
│   └───FLASH
├───DICT
├───ACCOUNTS
│   └───1987
├───BATCH
├───DOS
│   └───EGA
├───BUSINESS
├───BASIC
│   ├───ALLIANCE
│   ├───COMPILER
│   └───GRAPHICS
├───GAMES
├───LOTUS
├───WORDSTAR
│   └───STARINDX
└───XLAT

C:\>_
```

Figure 90: Output of the Tree command in DOS 4.0

you can see, ChkDsk produces a much more compact listing than does Tree in releases before 4.0. On the other hand, ChkDsk lists files in subdirectories of a directory as soon as the subdirectory name appears in the directory. Thus, files in a subdirectory may appear among the files in their parent directory. The \DBASE\FLASH file names appearing among the \DBASE file names in the ChkDsk entry illustrates this. You may find either preferable, depending on your needs.

To produce a file containing all the file names on your disk using DOS 2, you can either use ChkDsk instead of Tree, or enter the following two commands:

```
dir \>catalog.hd
tree /f>>catalog.hd
```

See Redirection and Pipes for more information about these commands.

```
C:\>tree /f

Directory PATH listing for Volume JK'sTurboAT
Volume Serial Number is 2C06-003C
C:.
|    COMMAND.COM
|    CONFIG.WIN
|    CONFIG.DV
|    CONFIG.SYS
|    ANSI.SYS
|    ANSI43.SYS
|    DM.SYS
|    DRIVER.SYS
|    MSMOUSE.SYS
|    RAMDRIVE.SYS
|    X2CDRIVE.SYS
|    EMM.SYS
|    X2CEMM.SYS
|    AUTOEXEC.BAT
|    CAROUSEL.OPT
|    LASTBKUP
|    MIRROR.FIL
|    SPOOL.COM
|    TREEINFO.NCD
|
|---DBASE
|       ASSIST.HLP
|       DBASE.COM
|       DBASE.LD1
|       DBASE.MSG
|       DBASEINL.OVL
|       HELP.DBS
|       CONFIG.DB
|       DBFIX.EXE
|       DBLOG.FIX
|       DBLOGM.FIX
|       DBASE.EXE
|
|---FLASH
|           FL.COM
|           FLCLIP.OBJ
|           FLDEFALT.SYS
|           FLEDITOR.000
|           FLEDITOR.CHN
|           FLASHUP.COM
|           FLGEN.DB2
|           FLGEN.DB3
|           FLHELP
|           FLHLPWIN
|           FLINSTAL.CHN
|           FLMAIN.CHN
|           FLMAKDB2.000
|           FLMAKDB2.CHN
|           FLMAKDB3.000
|           FLMAKDB3.CHN
|           FLMAKHLP
|
.
.
.
```

Figure 91: Partial output of the Tree /f command in DOS 4.0

```
C:\>tree /a

Directory PATH listing for Volume JK'sTurboAT
Volume Serial Number is 2C06-003C
C:.
+---DBASE
|   \---FLASH
+---DICT
+---ACCOUNTS
+---BATCH
+---DOS
|   +---EGA
+---BASIC
|   +---ALLIANCE
|   +---COMPILER
|   \---GRAPHICS
+---GAMES
+---LOTUS
+---WORDSTAR
|   \---STARINDX
|   \---INSET
\---XLAT

C:>_
```

Figure 92: Output of the Tree /a command in DOS 4.0

TSRs

Overview

Application programs are loaded into memory when you invoke them. Most programs abandon the portion of memory they use when you leave them, and are called *transient* programs. Other programs, however, leave a portion of their code in memory, and are called *terminate-and-stay-resident* programs (TSRs for short).

TSR programs generally intercept the data passing through your computer and modify it in some way. The following DOS commands invoke TSRs:

COMMAND	FUNCTION
Append	Adds information to the DOS environment regarding directories to search for nonexecutable files
DOSShell	Places the DOS 4.0 file-management and menu system in memory, or places a small core portion of that program in memory, so that it can be recalled after you use an application program.
FastOpen	Manages a buffer in which file names and path names referenced in commands are stored. (This reduces the time it takes to locate and open files you have already opened.)

COMMAND	FUNCTION
GrafTabl	Maintains a table of high-order characters in video memory for use with graphics adapters.
Graphics	Modifies output sent from the screen to the printer so that the printer can print graphics screens.
Mode	(When used to redirect printer output) intercepts output intended for a parallel port and sends it to a serial port.
NLSFunc	Maintains a table, in memory, of information needed to switch among international character sets.
Print	Maintains a list of files to be printed and keeps other text from going to the printer while those files are being printed.
Share	Maintains a table of files on your computer being shared on a network and manages access to your files by other computers on the network.

When loaded into memory, these programs reduce your available RAM and cannot be removed except by rebooting the computer. GrafTabl, in DOS versions 3.2 and earlier, loads another copy of itself into memory each time you invoke it, thereby taking up additional memory each time.

The coding that made these programs possible has led to the development of many TSR utility programs. The first, and probably the best known, was SideKick, a program that puts a notepad, appointment calendar, calculator, ASCII table, and phone dialer into memory. You access these programs by pressing a *hot key*—generally a key combination not normally used, such as Ctrl-Esc. These programs remain dormant until you press the hot key, then they appear on the screen for your use, disappearing into the background when you tell them to.

Since Microsoft never published standards for such programs, unorthodox programming tricks have been used to compensate for DOS's limitations. As a result, the programs may conflict either with application programs or with each other. When using TSR utilities, you may need to experiment extensively to be sure that the TSR utilities are loaded in an order that minimizes conflict.

Some TSR utilities can be removed from memory. However, you should always remove them from memory in the order opposite that in which you loaded them. When you load several, they are "stacked" in memory, one on top of another. If you remove something from somewhere other than the top of the stack, the whole pile will probably crash and lock up your computer.

For the same reason, never load TSR utilities while a transient program is running. If you do so, there will be a hole in the stack when you end the transient program, and you will face similar dire consequences.

Tips

See Append, DOSShell, FastOpen, GrafTabl, Graphics, Mode, NLSFunc, Print, and Share for information on DOS TSRs. See Install= for another method of placing TSRs in memory.

Type

Syntax

type *filename.ext*

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 1.0 to 3.3
IBM DOS 4.0

Overview

The Type command displays text files on the screen. It displays all the characters you can type on the keyboard (ASCII codes 32 through 127), and all the high-order ASCII characters (ASCII codes 128 through 254). (See ASCII Code.) It also displays some of the control characters as their graphics equivalents. Table 24 shows both the control characters that Type cannot display and their effect on the display, if any.

The Type command can optionally print files on the printer.

Procedure

To display a text file on the screen, enter the command

```
type filename.ext
```

where *filename.ext* is the name of the file you want to examine. You cannot use wildcard characters in the file name (see Wildcard Characters). If you do so, you will see the message

```
File not found
```

If the file is longer than a single screen, the text will scroll until the end of the file is reached, at which point the DOS prompt reappears. To halt the display, press Ctrl-S or Ctrl-Num Lock. To restart the display, press any key.

TABLE 24 **Control Characters Not Displayable with the Type Command**

Ctrl-@	The next character will appear at the beginning of the following line
Ctrl-G	Sounds a beep
Ctrl-H	Backspaces one column, erasing the previous character
Ctrl-I	The next character will appear at the next eight-column tab stop following a Ctrl-I
Ctrl-J	Line feed
Ctrl-M	Carriage return
Ctrl-P	Displays a space
Ctrl-Z	No characters displayed after a Ctrl-Z is encountered
Ctrl-[	No effect

There are two ways to print a file using the Type command. If you want to see the file on the screen as you print it, press Ctrl-P or Ctrl-PrtSc before entering the Type command. (See Command Keys.) Press the same key combination after you are through to stop the echoing of the screen to the printer.

If you do not want to see the text on the screen as you print it, enter a command of the form

```
type filename.ext>prn
```

(See Device Names and Redirection and Pipes.)

Examples The Type command can display extensive text quickly. This technique is most useful for displaying menus or help screens from batch files (see Batch Files). You can also use the Type command to send ANSI escape sequences in a file to the screen or keyboard (see ANSI.SYS). For an example of these techniques, see TSR.BAT and TSR.MNU in the ANSI.SYS entry. For comparison, Figure 93 shows an all-text version of the TSR.MNU menu screen, without the ANSI escape sequences. You will make your work easier when creating this program if you first type in this text screen, then add the ANSI escape sequences. To display this screen, simply enter

```
type tsr.mnu
```

```
                    CONFIGURATION MENU

[F1]  Help with this menu

[F2]  DOS GRAPHICS Utility

[F3]  Software Carousel

[F4]  Microsoft Windows

[F5]  Keyboard Enhancer only

[F6]  Keyboard Enhancer, Spooler, and Desktop Organizer

[F7]  Keyboard Enhancer, Spooler, Desktop
       Organizer, and Outliner
[F8]  Keyboard Enhancer, Spooler, Desktop
       Organizer, and Thesaurus

[F10] No Programs in the Background
```

Figure 93: TSR.MNU—Text for screen display

Warnings

The Type command cannot display program or other binary files because it interprets each byte in a file as an ASCII code and displays the equivalent character. Thus, if you try to display a file of this type, you will see a mess of gibberish accompanied by random beeps.

The Type command interprets every byte with a value of 7 as a Ctrl-G, which sounds a beep. It interprets the first byte with a value of 26 as a Ctrl-Z, or end-of-file mark, and immediately halts the display, returning you to the DOS prompt.

Some word processing programs, such as WordStar, replace some characters that you enter with their high-order equivalents (that is, with the graphics character having an ASCII code 128 higher than the normal character). This is how these programs include formatting information in their files. You can display such files with the Type command, but they will be barely readable. However, you will probably be able to understand enough of the text to determine if what you are looking for is in the file.

Tips

To display a file one page at a time, enter the command

```
type filename.ext|more
```

or the command

```
more<filename.ext
```

To display more than one file on the screen at a time, copy them to the console instead of using the Type command. Enter a command of the form

```
copy filename.ext con
```

where *filename.ext* includes a wildcard pattern (see Wildcard Characters). To display a series of files a screen at a time, use the More filter:

```
copy filename.ext¦more>con
```

For related information, see Filters, More, and Redirection and Pipes.

USER. *See* Device Names

USER is an alternative name for COM1 in some versions of DOS.

VDISK.SYS

Syntax

{1} `device=vdisk.sys [size] [sector-size] [dir-size] [/e]`
{2} `device=vdisk.sys [size] [sector-size] [dir-size] [/x]`

DOS Versions

{1} Some MS-DOS versions 2.1 to 3.31
{1} PC-DOS 3.0 to 3.3
{all} MS-DOS 4.0
{all} IBM DOS 4.0

Overview

VDISK.SYS is a device driver that creates a *RAM disk*—a portion of RAM that functions as an extra disk drive. You load VDISK.SYS by means of a `device=` command in the CONFIG.SYS file (see CONFIG.SYS File). This driver allows you to specify the total size of the disk, the sector size, and the number of directory entries. If you have a PC AT with extended memory (memory beyond 640K), or an 386 PC, you can place the RAM disk in extended memory using the /e switch. The device driver itself uses only about 1K of system memory.

V VDISK.SYS

With DOS 4.0, you can alternatively place the RAM disk in EMS (expanded) memory using the /x switch, if you have one of three memory cards manufactured by IBM and you configure the XMA2EMS.SYS device driver correctly. (If you have a 386 PC, you will also need the XMAEM.SYS device driver. See Memory, XMAEM.SYS, and XMA2EMS.SYS for further information.)

You can increase the speed of many disk I/O operations by placing frequently read files on a RAM disk. Disk I/O on a RAM disk involves only moving data electronically from one memory location to another. With any other type of disk drive, the speed of the moving parts limits I/O speed.

A second advantage of RAM disks is that they cannot be removed from the computer. If you do not have a hard disk, you can use a RAM disk to store files that you want available at all times. (See AUTOEXEC.BAT File.)

RAM disks have two notable disadvantages. First, they take up some memory. If you run large programs and have no expanded or extended memory in your system, a RAM disk requires careful planning (see Memory). Your RAM disk must be large enough to be useful and still leave enough memory for your programs to run. If you do have extended or expanded memory, you can place your RAM disk in this type of memory and use a smaller amount of conventional memory.

The second disadvantage of RAM disks is that they are *volatile*—that is, they disappear, along with everything on them, if the power goes off even for an instant. They also disappear whenever you reboot your computer. Therefore, you should back up your modified files from a RAM disk.

Procedure

To create a RAM disk with VDISK.SYS, add a line of the following form to your CONFIG.SYS file:

```
device=vdisk.sys [size] [sector-size] [dir-size] [/e]
```

Table 25 summarizes the values to enter in this command.

If your disk is in conventional memory, you can specify a *size* up to 64K less than your total conventional memory. (If you have 64K or less of conventional memory, you cannot install VDISK.) If you want a disk of the maximum size, you should specify a sector size of 512; otherwise, VDISK adjusts the sector size to 512.

In addition to sector sizes of 128 and 512 bytes, you can also have sectors of 256 bytes. The larger the sector size, the more data can be read into memory at a time (see Disk Sectors).

Dir-size refers to the maximum number of entries in the root directory. Since directory entries are allotted in sectors, you can limit the

TABLE 25 **VDISK.SYS Parameter Values**

PARAMETER	UNIT	MINIMUM	MAXIMUM	DEFAULT
size	kilobytes	1	32,768	64
sector-size	bytes	128	512	128
dir-size	entries	4	512	64
transfer-size	sectors	1	8	8

root directory to four entries only if you use a sector size of 128 (because each directory entry takes 32 bytes) (see Directories). The actual number of usable entries will be one less than *dir-size* because DOS gives the disk a volume label that uses one entry (see Label).

To place the disk in extended memory, use the /e switch. If you use the /e switch, you can also specify a *transfer size*—the number of sectors to be transferred at one time into conventional memory with a command of the form /e:*n* where *n* is the number of sectors to transfer. If you do not specify this number, the default is 8. If you find that you are losing data, decrease the default until all data is transferred correctly.

You can add comments between the parameters to VDISK.SYS in your CONFIG.SYS file. However, they will not appear on the screen when the driver is loaded. You cannot include carriage returns, Ctrl-Js (line feeds), or slashes in your comments.

To create more than one RAM disk, add additional `device=vdisk.sys` entries to your CONFIG.SYS entry. Each RAM disk you load will receive the next available drive name. If you have a hard disk, the first RAM disk will be drive D, the next drive E, and so on. If you do not have a hard disk, the first RAM disk will be drive C, and so on. Even if you have only one drive, your first RAM disk will be drive C. A and B are drive names reserved for diskette drives.

When DOS loads the device driver, it displays a message of the following form:

```
VDISK Version 3.xx virtual disk C:
Buffer size:          64 KB
Sector size:         128
Directory entries:    64
```

The entries shown are for the default values on a system with no hard disk.

Examples

RAM disks can be a special convenience when you have extensive memory and do not have a hard disk since they give you a nonremovable drive, plus the advantage of increased I/O speed.

To create a 64K RAM disk in conventional memory, with a sector size of 128 bytes and 64 root directory entries, use the command `device=vdisk.sys` in your CONFIG.SYS file. You might want such a RAM disk if you have 640K of conventional memory, no extended memory, and you use relatively large applications. DOS will take up about 40K, and the RAM disk and its driver about 65K, leaving you about 535K to run programs. You will have even less space if you have anything else in your CONFIG.SYS file, or if you use resident utilities (see TSRs). A RAM disk of this size is large enough to hold several batch files, a few essential DOS external programs, and/or an overlay file for WordStar.

You would need a RAM disk of at least 260K if you want to place the overlay file for dBASE III PLUS on a RAM disk. You can save some space on the RAM disk by reducing the number of directory entries. To use a RAM disk with this application, you might use the command

```
device=vdisk.sys 266 512 8
```

This would leave you 333K to run dBASE (enough, although less than ideal). However, the time you would save by reading the overlay from the RAM disk might make up for the inefficiency of running in limited memory.

To create a 1-megabyte RAM disk in extended memory, you might use a command of the form

```
device=vdisk.sys 1024 512 /e
```

This would give you a very large RAM disk. You could copy entire applications to it from your hard disk (or even from diskettes) and run them from it. You can also save time with editing and word processing programs by placing the files to be edited on the RAM disk. (This will not help, however, if your word processor keeps the entire file in memory.) Once you set it up, a large RAM disk in extended memory can almost substitute for a hard disk. Just remember to back up all your modified files every ten minutes or so. See AUTOEXEC.BAT File for examples of ways to set up a RAM disk once you have configured it.

Warnings

RAM disks disappear if the power goes off even for a moment. It's therefore very important to back up your files frequently from a RAM disk to a less volatile medium.

The defaults described in this entry are those used by PC-DOS. If you have a version of VDISK.SYS that came with another version of DOS, its parameters may have different defaults or accept different ranges of values. It may also allow you to install a RAM disk on an expanded memory card. Check your DOS manual for complete information.

Do not use VDISK.SYS with Microsoft Windows. To use a RAM disk with that program, use only the RAMDRIVE.SYS device driver included in the package.

Tips

If your application program includes a macro facility, or if you use a keyboard macro program, you can design a macro to first save a file in memory to the RAM disk and then copy it to another disk. If you don't have access to keyboard macros, create batch file such as RAMAPP.BAT, which follows. This is a generic form that lets you load any application from a diskette in drive A, and any data files from a diskette in drive B. It automatically saves any modified date files to drive B. It does not save new files that you create on the RAM disk.

RAMAPP.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`c:`	Makes the RAM disk current
`copy a:%1`	Copies the program file from drive A
`copy b:%2 %3 %4 %5 %6 %7 %8 %9`	Copies up to 8 data files from drive B
`%1 %2`	Invokes the application, with the first data file as a parameter
`echo ^GPlace original data disk in drive B`	Reminds you to be sure the right disk is in drive B
`pause`	Pauses for keystroke, with message
`replace c:*.* b:`	Replaces all files that were on original data disk.

To invoke this program, enter a command of the form

`ramapp` *`program datafile.ext datafile.ext . . . datafile.ext`*

where *program* is the name of you application, and the iterations of *datafile.ext* are the names of the data files to copy to the RAM disk. If your application will open a file whose name is entered as a command-line parameter, be sure that the first data file you name is the one you want to open.

You can modify the RAMAPP file for your own purposes by doing the following:

- Changing the drive letters to ones appropriate to your system
- If your application uses standard extensions for its data files, changing the last line to read `xcopy c:*.ext b:` where *ext* is the standard extension
- Building in the name of the application program in place of the %1 parameter, and changing the name of this batch program appropriately

For example, if you use WordStar 4.0 and have a large enough RAM disk, you could modify the file so that it looks like RWS.BAT, which follows.

RWS.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`c:`	Makes the RAM disk current
`copy a:ws*.*`	Copies the program files from drive A
`copy b:%1 %2 %3 %4 %5 %6 %7 %8 %9`	Copies up to 9 data files from drive B
`ws %1`	Invokes WordStar, and tells it to open the first data file copied
`echo ^GPlace original data disk in drive B`	Reminds you to be sure the right disk is in drive B
`pause`	Pauses for keystroke, with message
`replace c:*.* b:`	Replaces all files that were on original data disk

Some MS-DOS versions include a RAM disk driver called RAMDRIVE.SYS in place of VDISK.SYS (see RAMDRIVE.SYS).

If you have an expanded memory board, but not PC AT or 386 extended memory, you can probably use a driver packaged with the board to create a RAM disk in expanded memory. If you do so, be sure to load the driver after loading the driver that initializes the memory board.

Ver

Syntax

```
ver
```

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

The Ver command indicates which version of DOS you are using.

Procedure

To see which version of DOS is currently running your computer, enter the command

```
ver
```

You will see a message appropriate to the release of DOS that you booted your system with, such as one of the following:

```
Compaq Personal Computer DOS Version 3.31

IBM Personal Computer DOS Version 3.30

MS-DOS Version 3.20

ITT-DOS Version 2.11
```

Tips

If you use several versions of DOS and you use an AUTOEXEC.BAT file, it's a good idea to include the Ver command in your AUTOEXEC.BAT files (see AUTOEXEC.BAT File).

Verify

Syntax

{1} verify on
    verify off
{2} verify

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

Although errors in copying files are very rare, the Verify command slightly reduces the risk of such errors. When you use the Verify option, DOS reads each sector's checksum after writing to the sector, then compares it to the checksum of the data that was written.

Verify does not read the data after writing it nor compare it to the original, as you might expect. Verify simply ensures that the data is not written to a bad sector. Of course, the disk drive controller normally performs this function whether you ask it to or not. Thus, using the Verify command takes some extra time, and doesn't accomplish very much.

Procedure

Setting Verify Status

{1} verify on
    verify off

To have DOS perform the verification procedure, enter the command

```
verify on
```

After you enter this command, DOS verifies every copy you make until you enter a

```
verify off
```

command. If you do not enter a Verify on command, DOS will not verify copies.

Checking Verify Status

{2} `verify`

To check whether you have asked for verification, enter the Verify command with no parameter. You will see one of the following messages:

```
VERIFY is on
```

or

```
VERIFY is off
```

Tips

The Verify command does the same thing as the /v switch to the Copy and Xcopy commands (see Copy and Xcopy).

A much more reliable way of ensuring the integrity of your data is to run Comp or FC on the copy after you make it (see Comp and FC).

There is no Verify option for copying diskettes. To verify a copy made with DiskCopy, use the DiskComp command (see DiskCopy and DiskComp). Unlike Verify, DiskComp actually compares the data on the source and the target.

Vol

Syntax

`vol [`*drive*`]`

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

Since DOS 2.0, it has been possible to enter a disk name, or volume label, on each disk with the /v switch to the Format command (see Format). Since DOS 3, you could also enter or change a volume label with the Label command (see Label). The Vol command allows you to see the volume label on a disk or to verify that there is none.

Procedure/ Examples

To examine the volume label on a disk, enter the Vol command followed by the name of the drive containing the disk whose label you want to see. You will see a message of one of the following types:

```
Volume in drive A is ACCOUNTS'88
```

or

```
Volume in drive C has no label
```

Tips

If you use many diskettes, it's much simpler to keep track of them if you give them volume labels. The volume labels allow you to find out which disk is in a drive using the Vol command, instead of viewing a complete directory.

Wildcard Characters

DOS Versions

MS-DOS 2.1 to 4.0
PC-DOS 2.0 to 3.3
IBM DOS 4.0

Overview

Wildcard characters are special characters that let you issue commands affecting groups of files. (See Command Characters.) The wildcard characters are the question mark and the asterisk. The question mark substitutes for any single character at a particular position in a file name. The asterisk substitutes for the character in the position it occupies, and all remaining characters.

Not all commands allow you to use wildcard characters; some require a specific file name.

There are two different sets of rules for wildcard characters, one for the Dir command and another for all other commands that accept file names with wildcard characters. Let's review the more general rules first.

Procedure/ Examples

When using wildcard characters, you must treat the file name and the extension as separate entities; that is, if you enter a command containing a file name such as

```
clink*
```

it will affect only files whose names begin with the characters C L I N K, but that have no extension. If you want to include all files whose names began with these characters, you must use the file name

```
clink*.*
```

You can refer to all files with a given extension—.EXE, for example—by using the form

```
*.exe
```

Similarly, you can refer to all files with the same name but different extensions with a form such as

```
reports.*
```

You cannot substitute an asterisk character for several characters in the middle of a file name. Thus, DOS understands

```
re*ts.*
```

as being the same as

```
re*.*
```

because DOS translates the first asterisk character to the number of question-mark characters needed to complete the file name or extension. If you issue a command containing a file name such as either of the above, and you see an error message, it may include the form

```
RE??????.???
```

filling out the file name to the full eight characters, and the extension to three.

Question-mark characters in a file name substitute for any single character that occurs at the position they occupy in any file name that matches the rest of the pattern you enter. The question mark also substitutes for a nonexistent character at the position it occupies. Let's clarify this with a few examples.

Suppose you had a directory including the following file names:

CLUNKERS.TST
CLINKER.BAS
CLINKER.TXT
CLUNKER.TXT
ALINK.TST
BLINKS.TMP
CLINK.BST
CLINK
CLINK.2

CLINK.21
CLINK.213

If you enter a command including the file name

```
?link.*
```

the command acts on the files ALINK.TST, CLINK.BST, CLINK, CLINK.2, CLINK.21, and CLINK.213. However, if you enter

```
?link?.*
```

it also affects BLINKS.TMP. If you enter the command with the file name

```
?link?.?
```

the command acts on CLINK and CLINK.2.

To take another example, if your command includes the file name

```
?l?nker?.*
```

it acts on CLUNKERS.TST, CLINKER.BAS, CLINKER.TXT, and CLUNKER.TXT.

Perhaps the clearest way to understand the effect of the question mark is to visualize the way the file names appear in a directory. As you know, they do not appear closed, with a period between the name and the extension. Rather, they appear in columns, with eight columns allotted for the file name, one for a separator (which appears on the screen as a space), and three for the extension:

```
CLUNKERS TST
CLINKER  BAS
CLINKER  TXT
CLUNKER  TXT
ALINK    TST
BLINKS   TMP
CLINK    BST
CLINK
CLINK    2
CLINK    21
CLINK    213
```

If you look at the entry on Directories, you will see that the extra spaces that you don't use in a file name actually appear on disk in the directory as space characters. Thus, the question mark substitutes for anything in its column as it appears in the directory. If your wildcard pattern ends in three question marks, the command will substitute three spaces for any file name that isn't long enough but otherwise matches, as Figure 94 shows.

```
D:\>xcopy ?l?nk???.* d:\test\

Reading source file(s)...
CLUNKERS.TST
CLINKER.BAK
CLINKER.TXT
CLUNKER.TXT
ALINK.TST
BLINKS.TMP
CLINK.BST
CLINK
CLINK.1
CLINK.21
CLINK.213
       11 File(s) copied

D:\>_
```

Figure 94: The effect of the question-mark character

You could get rid of all these files, and presumably no others, by entering the command

```
del ?l?nk*.*
```

In the Dir command, DOS gives you a lot more leeway. An asterisk character replaces everything that follows it, including the extension. Thus,

```
dir cli*
```

displays the directory listing for CLINKER.BAS, CLINKER.TXT, CLINK.BST, CLINK, CLINK.2, CLINK.21, and CLINK.213. For further details on wild-cards with the Dir command, see Dir.

Tips

The question-mark character is especially useful if you adopt a naming convention that uses each column in a file name to represent a specific type of information (see File Names).

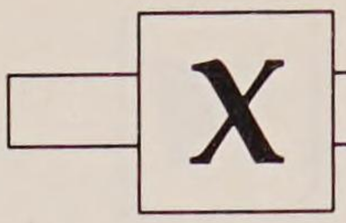
Xcopy

Syntax

`xcopy` *`source[filename.ext] [target][filename.ext] [/switches]`*

DOS Versions

MS-DOS 3.2 to 4.0
PC-DOS 3.2 to 3.3
PC-DOS 4.0



XCOPY

Overview

Xcopy is an extended copy command that allows you to copy files much more quickly than the Copy command and to select the files to copy. You can select by name, by date, or by archive status. Xcopy can copy entire directories, including subdirectories subordinate to them. If the subdirectories do not exist on the target, Xcopy will create them. Xcopy can optionally create subdirectories to match those on the source, even if some of them contain no files to copy.

Xcopy is fast because it reads as many files as memory will hold before it begins writing to the target. When it writes, it writes all the files in memory. In contrast, the Copy command reads one file at a time into memory and then writes it. The difference in time is more noticeable when one of the disks involved is a diskette, and even more remarkable when you copy files from one diskette to another in a single drive. In the latter situation, Copy requires you to swap diskettes after reading each file and again after writing each file.

The power of Xcopy lies in its many switches, which provide numerous ways to select the files to copy. These are the switches and their effects:

SWITCH	EFFECT
/a	Copies only files with the archive bit set. Does not clear the archive bit of either the original or the copy. (See ATTRIB and File Attributes for more information about the archive bit.)
/m	Copies only those files whose archive bit is set (that is, those that have been modified since they were last backed up). This switch clears the archive bit of the source.
/d:*date*	Copies only files created or changed on or after the specified date. Uses the date format for the country to which your computer has been set. (See Country = or Select for more information about selecting date formats.)
/s	Copies files from the subdirectories of the specified source as well as the source directory, creating equivalent subdirectories on the target if there are any files matching your selection criteria in the source subdirectories.
/e	Creates subdirectories on the target to match those on the source. However, if the selection criteria you used result in no files copied from a given subdirectory, the matching subdirectory on the target will be empty.
/v	Performs checksum verification while copying (see Verify).
/p	Displays each file name before copying, requesting your permission to continue by displaying a message of the form *SOURCE\FILENAME.EXT* (Y/N)? This gives you the opportunity to exclude files that may match your criteria but that you don't want on the target.

SWITCH	EFFECT
/w	Displays the message `Press any key to begin copying file(s)` and waits for a keypress. This gives you the opportunity to switch diskettes in a drive if you have read the Xcopy program from a diskette and want to use the same diskette drive as either the source or the target.

Procedure/ Examples

Xcopy is an external command; using it requires having the DOS program XCOPY.EXE on a disk in your computer. If the program is neither in the current directory nor on the search path, preface the command with a drive name and/or path name; for example,

```
c:\dos\xcopy c: b:
```

To copy files using the Xcopy command, you must specify the source. If the target drive is current, you may specify only the source. If the source is current, you must also specify a target. If neither is current, specify both.

The screen will then display the message

```
Reading source file(s) . . .
```

Xcopy reads as many of the specified files as fit in available memory, then displays the name of each source file as it is copied. If your command specifies more files than can fit in memory, Xcopy then reads more source files, displaying the same types of messages.

Copying Complete Directories To copy all the files in the current directory on drive C to another directory on drive C called \REPORTS, you enter either

```
xcopy c: \reports\
```

or

```
xcopy . \reports\
```

If you do not place the additional backslash after the name of the target directory, you will be asked

```
Does REPORTS specify a file name
or directory name on the target
(F = file, D = directory)?
```

If you press F, each file in the source will be copied to a file called REPORTS in the root directory. When the command has finished executing, REPORTS will be a copy of the last file in the source directory. If you press D and the directory doesn't exist, Xcopy will create it.

To copy from the \REPORTS directory to the current directory, you need only specify the source:

```
xcopy \reports
```

Selecting Files by Name To select files by name, enter a file name or wildcard pattern as part of the source:

```
xcopy *.rpt \reports
```

If you do not want to copy some matching files, add the /p switch:

```
xcopy *.rpt \reports /p
```

If you want the copies to have names that are different from the originals, specify a target file name:

```
xcopy *.rpt \reports\*.bak
```

You can use this technique to make second copies of a group of files in the source directory:

```
xcopy *.rpt *.bak
```

Either of these commands will overwrite any files in the target directory with the same file name and the .BAK extension.

Selecting Files by Date Suppose today were Friday, November 24, 1989, and you wanted to make a backup copy of all files created or changed this week. Assuming the source is current, you might enter the command

```
xcopy ./d:11/20/89 a:
```

This command will copy only files from the source dated November 20, 1989, or later. However, if the combined files are larger than the target diskette in drive A, you will see the message

```
Insufficient disk space
nn File(s) copied
```

and Xcopy will quit. At this point, there is no easy way to select the remaining files and exclude those you have already copied. However, if you have enough room on your hard disk, you can accomplish this by using an intermediate temporary directory. The batch file DATEBAK.BAT, which follows, will do this for you.

DATEBAK.BAT

COMMANDS	COMMENTS
`echo off`	Suppresses display of commands
`xcopy %1 \temp\ /d:%2`	Copies all files created or changed since the date you specify on the command line to the \TEMP directory, creating the directory in the process
`cd \temp`	Makes the \TEMP directory current
`:nextdisk`	Beginning of copy loop
`echo ^GInsert backup diskette.`	Message with a beep
`pause`	Waits for keystroke
`xcopy . a: /m`	Copies files whose archive bit is set to drive A, clearing the archive bit after copying
`if errorlevel 4 goto nextdisk`	Repeats the loop starting from the `:nextdisk` label if Xcopy terminates for any reason such as insufficient space on the target. (See ErrorLevel for a table of exit codes and an explanation of this command.)
`cd\`	Ensures that the \TEMP directory is no longer current so that it can be deleted
`echo y|del \temp`	Deletes all files on \TEMP directory; Echo command sends a y character to the Del command in response to the `Are you sure (Y/N)?` message. (See Redirection and Pipes for details.)
`rd \temp`	Removes the \TEMP directory

The trick in this program is to use the Xcopy command twice. The command is used twice because Xcopy command will terminate if the selected files take up more than a single diskette. If you try to continue with another diskette, the same files will be copied again because Xcopy won't know which files it has already copied.

The strategy, therefore, is to create a directory containing only those files to be backed up and to use the /m switch to keep track of which ones have been copied. To create this directory, use the command

```
xcopy %1 \temp\ /d:%2
```

This command uses the /d switch and creates a new directory containing only the relevant files. In the process, it set the archive bit on all of these files. When the command

```
xcopy . a: /m
```

copies these files, it will clear their archive attribute. This means that each time the loop reaches the second Xcopy command, only files that have not yet been copied—that still have their archive attribute set—will be copied.

The second trick in this program is to test Xcopy's exit code with the ErrorLevel modifier (see ErrorLevel). The exit code of 4 indicates an abnormal termination. Therefore, when the command terminates before copying all selected files, the program cycles through the loop again. (See GoTo.) When all remaining files are copied, Xcopy terminates normally with an exit code of 0, and execution continues past the loop.

Finally, at the end of the program, the temporary directory is deleted to keep from cluttering up your hard disk.

For another program that copies files using Xcopy when the target diskette is not large enough to hold all the files, see DIRBAK.BAT, in the Hard Disks, Backing Up and Restoring, entry.

Warnings

If you execute the Xcopy command from a batch file, pressing Ctrl-Break terminates the Xcopy command, but not the batch file. You must press Ctrl-Break twice to terminate the batch file while Xcopy is executing.

You cannot use the /e and /m switches in the same Xcopy command. All other switches are compatible.

Tips

To copy only files that exist on the target or only files that do not exist on the target, use the Replace command (see Replace).

The Xcopy command has exit codes that you can test with the ErrorLevel modifier of the If command. See ErrorLevel for a table of exit codes.

For related information, see Verify, Copy, Replace, and Hard Disks, Backing Up and Restoring.

XMAEM.SYS

Syntax

```
device=xmaem.sys [pages]
```

DOS Versions

MS-DOS 4.0
IBM DOS 4.0

Overview

XMAEM.SYS is a special device driver to let IBM PS/2 computers using the 80386 chip take advantage of extended memory (see Memory). Although the 80386 chip can address 16 gigabytes of memory, DOS can address only 640K directly. XMAEM.SYS makes extended memory available to DOS by emulating expanded memory—specifically a board called the IBM Personal System /2 80286 Expanded Memory Adapter /A. When XMAEM.SYS is installed, it causes the 80386 chip to treat the memory installed on an IBM PS/2 80286 Expanded Memory Adapter /A board as expanded memory meeting the LIM-EMS 4.0 specification.

Procedure

XMAEM.SYS is a device driver installed via the CONFIG.SYS file. To use it, place the command `device=xmaem.sys` in your CONFIG.SYS file. If the device driver is not in the root directory of your boot disk, you can specify the drive and path where it is located; for example,

```
device=c:\dos400\xmaem.sys
```

If you don't include a size parameter, XMAEM.SYS will treat the all the memory on the IBM PS/2 80286 Expanded Memory Adapter /A board as expanded memory. If you want to divide the memory between extended and expanded memory, you must specify the number of 16K pages of memory to be used as expanded memory. The minimum number of pages is sixty-four, which results in 1,048,576 bytes (or 1 megabyte, calculated as 1024 × 1024) of expanded memory.

Having used the XMAEM.SYS driver to turn your extended memory into expanded memory, you must then install the XMA2EMS.SYS device driver to manage the expanded memory. See XMA2EMS.SYS for details.

Warnings

If you have a 386 PC that is not also a PS/2 and/or a memory board that isn't an IBM PS/2 80286 Expanded Memory Adapter/A board, you will see the message

```
Illegal command in CONFIG.SYS file, line nn
XMAEM.SYS not loaded
```

where *nn* is the line containing the command that loads the driver.

XMA2EMS.SYS

Syntax

{1} `device=xma2ems.sys frame=hhhh [p254=hhhh] [p255=hhhh] [/x:pages]`

{2} `device=xma2ems.sys p0=hhhh p1=hhhh p2=hhhh p3=hhhh [p254=hhhh] [p255=hhhh] [/x:pages]`

DOS Versions

{**all**} MS-DOS 4.0

{**all**} IBM DOS 4.0

Overview

XMA2EMS.SYS is a device driver used to manage expanded memory in DOS 4.0 if you have one of the following expanded memory boards:

- The IBM 2-megabyte Expanded Memory Adapter
- The IBM PS/2 80286 Memory Expansion Option
- The IBM PS/2 80286 Expanded Memory Adapter/A

If you have an IBM PS/2 computer with an 80386 chip, you can use XMA2EMS.SYS with its extended memory, or with the latter board, if you first install XMAEM.SYS. See Memory for a full explanation of these terms, and see XMAEM.SYS for an explanation of that device driver.

If you install this device driver successfully, you can install the DOS 4.0 VDISK.SYS RAM disk in expanded memory, and can use the /x switch to

the Buffers= and FastOpen commands to create their buffers in expanded memory, saving more conventional memory for your application programs (see VDISK.SYS, Buffers=, and FastOpen).

Procedure

You install XMA2EMS.SYS by using a Device= statement in your CONFIG.SYS file; for example,

```
device=xma2ems.sys frame=d000
```

If the device driver is not in the root directory of your boot disk, you can specify the drive and path where it is located; for example,

```
device=c:\dos400\xma2ems.sys frame=d000
```

You must specify the address, as a four-digit hexadecimal number, of either a 64K *page frame*, or of four individual 16K *memory pages*. If you want to use expanded memory for a RAM disk, FastOpen buffers or disk buffers, you must also specify addresses for memory pages 254 and 255.

You can specify a frame with the `frame` parameter if you have 64K of contiguous memory addresses to spare in the 1-megabyte DOS address space. Otherwise, you have to specify the page addresses. You can also specify a maximum number of 16K pages using the /x switch. The minimum is four, or 64K. If you don't specify a size, XMA2EMS.SYS will take control of all your expanded memory.

So how do you know what addresses to plug into this command? You don't. You have to experiment. The extensive options allow you to configure the board so that it doesn't use memory addresses that other boards in your computer use. However, it's not easy to find out what memory addresses a board uses. You might begin by using the command

```
device=xma2ems frame=d000
```

which is the default page frame address for EMS memory. If nothing else is using the 64K block of addresses at that point, your board will install itself properly when you reboot, and you're finished. If something is using one or more of the 16K blocks in that address range, you'll see a message such as one of the following:

```
Specified page address conflicts with installed adapter
 at address hhhh
Possible 16KB page available at: hhhh
Possible 64KB frame available at: hhhh
```

If your screen moves slowly enough, dump it to a printer, so you can refer to the numbers in the messages. If not, try to write them down. You can stop the scrolling by pressing Ctrl-Num Lock.

If you see a suggested frame address, edit your CONFIG.SYS file to replace the number d000 with the suggested number. If you see only page addresses, you'll need to specify four pages, designated p0, p1, p2, and p3. Rewrite your installation command accordingly. When you're done, it might look something like this:

```
device=xma2ems.sys p0=d000 p1=d400 p2=dc00 p3=c400
```

Reboot the computer and repeat the procedure. If you see any more messages concerning conflicts, or possible free pages, repeat the procedure. If the board hasn't installed itself, and you see no more available page addresses among the messages, you'll have work with the hardware. Take out one of the other boards in your system, and reboot it. *Be sure your computer is turned off and disconnected from the power source before removing any boards.* Note any messages that appear. Replace the board, and repeat the procedure with another board.

If your memory board installs itself when one of the boards was removed, you know which board was causing the conflict. If you see a message indicating that some additional address space is available when one of the boards is removed, reconfigure the memory board to use that space. However, in either case, you'll have to do without the conflicting board. If each of several boards gives you sufficient additional address space, you can choose which board to remove. Sometimes life forces tough choices upon us.

Unfortunately, you may not be through yet. If you want to use expanded memory for buffers or RAM disks, you have to specify addresses for two more memory pages. You must give an address for page 254 to use expanded memory for RAM disks and FastOpen buffers, and page 255 to place the buffers created by the Buffers= command in expanded memory. If there were two free page addresses noted in your last pass, you can use them. If not, you'll have to either do without these conveniences or go back to shuffling boards.

Warnings

Do not try to experiment with these values if you don't have one of the required IBM memory boards. If by some chance you're lucky enough to install the driver with a non-IBM board and you use any of the /x switches, it's possible that your primary hard disk's file allocation table

will become permanently scrambled as soon as any other program accesses expanded memory.

If you want to use an application that meets the LIM-EMS 3.2 standard, you must specify a 64K frame. The earlier standard doesn't work with the smaller 16K pages.

Tips

You probably don't know what any of this means. You probably don't have to. The discussion of the LIM-EMS 4.0 standard in the Memory entry will help a bit.

If you use a 386 PC, either of two commercially available memory QEMM-386 and 386-to-the-Max, will do everything that XMA2EMS does. However, they will work with *any* extra memory in the computer—expanded or extended—and they configure themselves automatically.

Appendices

Appendix A: ANSI Escape Sequences

TABLE A.1 **ANSI Escape Sequences for Controlling the Cursor**

ESCAPE SEQUENCE	EFFECT
←[*r*A	Moves the cursor up *r* rows
←[*r*B	Moves the cursor down *r* rows
←[*c*C	Moves the cursor right *c* columns
←[*c*D	Moves the cursor left *c* columns
←[*r*;*c*f	Places the cursor in the *r*th row and *c*th column
←[*r*;*c*H	Places the cursor in the *r*th row and *c*th column
←[H	Places the cursor in the home position (upper left corner of the screen)
←[2J	Clears the screen, leaving the cursor in the home position
←[K	Erases the current line from the current position of the cursor to the end
←[s	Saves (records) the current position of the cursor on the screen
←[u	Places the cursor at the position it was in when the ←[s sequence was issued.

TABLE A.2

ANSI Escape Sequences to Set the Video Mode

ESCAPE SEQUENCE	VIDEO MODE
←[=h	40 × 25 black and white text
←[=l	40 × 25 black and white text
←[=0h	40 × 25 black and white text
←[=0l	40 × 25 black and white text
←[=1h	40 × 25 color text
←[=1l	40 × 25 color text
←[=2h	80 × 25 black and white text
←[=2l	80 × 25 black and white text
←[=3h	80 × 25 color text
←[=3l	80 × 25 color text
←[=4h	320 × 200 color graphics
←[=4l	320 × 200 color graphics
←[=5h	320 × 200 black and white graphics
←[=5l	320 × 200 black and white graphics
←[=6h	640 × 200 black and white graphics
←[=6l	640 × 200 black and white graphics
←[=14h	640 × 200 16-color graphics*
←[=14l	640 × 200 16-color graphics*
←[=15h	640 × 350 monochrome graphics*
←[=15l	640 × 350 monochrome graphics*
←[=16h	640 × 350 16-color graphics*
←[=16l	640 × 350 16-color graphics*
←[=17h	640 × 480 2-color graphics*
←[=17l	640 × 480 2-color graphics*
←[=18h	640 × 480 16-color graphics*
←[=18l	640 × 480 16-color graphics*
←[=19h	320 × 200 256-color graphics*
←[=19l	320 × 200 256-color graphics*

** Available with DOS 4.0 and EGA or VGA only.*

TABLE A.3 **Parameters for ANSI Video Attribute Escape Sequences**

FOREGROUND	COLOR	BACKGROUND	OTHER ATTRIBUTES	
30	Black	40	0	All attributes off
31	Red	41	1	High-intensity text
32	Green	42		
33	Yellow	43		
34	Blue	44	4	Underlined text
35	Magenta	45	5	Blinking text
36	Cyan	46		
37	White	47	7	Inverse video
			8	Concealed text

Appendix B: ASCII Tables

TABLE B.1 **ASCII Codes**

DECIMAL	HEXADECIMAL	DISPLAY	CONTROL	ASCII
0	0		^@	NUL
1	1	☺	^A	SOH
2	2	☻	^B	STX
3	3	♥	^C	ETX
4	4	♦	^D	EOT
5	5	♣	^E	ENQ
6	6	♠	^F	ACK
7	7	•	^G	BEL
8	8	◘	^H	BS
9	9	○	^I	HT
10	A	◙	^J	LF
11	B	♂	^K	VT
12	C	♀	^L	FF
13	D	♪	^M	CR
14	E	♫	^N	SO
15	F	☼	^O	SI
16	10	►	^P	DLE
17	11	◄	^Q	DC1
18	12	↕	^R	DC2
19	13	!!	^S	DC3
20	14	¶	^T	DC4
21	15	§	^U	NAK
22	16	–	^V	SYN
23	17	↨	^W	ETB
24	18	↑	^X	CAN
25	19	↓	^Y	EM
26	1A	→	^Z	SUB
27	1B	←	^[	ESC
28	1C	∟	^\	FS
29	1D	↔	^]	GS
30	1E	▲	^^	RS
31	1F	▼	^_	US
32	20	(space)		
33	21	!		
34	22	"		
35	23	#		

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	¦
125	7D	}
126	7E	~
127	7F	⌂
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	P_t

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
159	9F	*f*
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	a
167	A7	o
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	▒
177	B1	▒
178	B2	▒
179	B3	─
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	╗
187	BB	╝
188	BC	╜
189	BD	╛
190	BE	┐
191	BF	└
192	C0	┴
193	C1	┬
194	C2	├
195	C3	─
196	C4	┼
197	C5	╞
198	C6	

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	▉
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀
224	E0	α
225	E1	β
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	μ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	ϕ
238	EE	ϵ
239	EF	∩

Table B.1 continues

DECIMAL	HEXADECIMAL	DISPLAY
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	^{n}
253	FD	^{2}
254	FE	■
255	FF	

TABLE B.2 **Extended ASCII Codes**

Note: The number/cursor keys on the separate numeric keypad of the 102-key keyboard all have the extended ASCII code 0;0 or 0;224. The plus, minus, and * keys have the standard ASCII codes for the characters. These keys do have separate *keyboard scan codes*, a subject beyond the scope of this book.

CODE	KEY OR MEANING
0;3	Null (Ctrl@)
0;15	Shift-Tab
0;16	Alt-Q
0;17	Alt-W
0;18	Alt-E
0;19	Alt-R
0;20	Alt-T
0;21	Alt-Y
0;22	Alt-U
0;23	Alt-I
0;24	Alt-O
0;25	Alt-P

CODE	KEY OR MEANING
0;30	Alt-A
0;31	Alt-S
0;32	Alt-D
0;33	Alt-F
0;34	Alt-G
0;35	Alt-H
0;36	Alt-J
0;37	Alt-K
0;38	Alt-L
0;44	Alt-Z

Table B.2 continues

CODE	KEY OR MEANING
0;45	Alt-X
0;46	Alt-C
0;47	Alt-V
0;48	Alt-B
0;49	Alt-M
0;50	Alt-N
0;59	F1
0;60	F2
0;61	F3
0;62	F4
0;63	F5
0;64	F6
0;65	F7
0;66	F8
0;67	F9
0;68	F10
0;71	Home
0;72	Cursor-up
0;73	PgUp
0;75	Cursor-left
0;77	Cursor-right
0;79	End
0;80	Cursor-down
0;81	PgDn
0;82	Ins
0;83	Del
0;84	Shift-F1
0;85	Shift-F2
0;86	Shift-F3
0;87	Shift-F4
0;88	Shift-F5
0;89	Shift-F6
0;90	Shift-F7
0;91	Shift-F8
0;92	Shift-F9
0;93	Shift-F10
0;94	Ctrl-F1
0;95	Ctrl-F2
0;96	Ctrl-F3
0;97	Ctrl-F4
0;98	Ctrl-F5
0;99	Ctrl-F6

CODE	KEY OR MEANING
0;100	Ctrl-F7
0;101	Ctrl-F8
0;102	Ctrl-F9
0;103	Ctrl-F10
0;104	Alt-F1
0;105	Alt-F2
0;106	Alt-F3
0;107	Alt-F4
0;108	Alt-F5
0;109	Alt-F6
0;110	Alt-F7
0;111	Alt-F8
0;112	Alt-F9
0;113	Alt-F10
0;114	Ctrl-PrtSc
0;115	Ctrl-Cursor-left
0;116	Ctrl-Cursor-right
0;117	Ctrl-End
0;118	Ctrl-PgDn
0;119	Ctrl-Home
0;120	Alt-1
0;121	Alt-2
0;122	Alt-3
0;123	Alt-4
0;124	Alt-5
0;125	Alt-6
0;126	Alt-7
0;127	Alt-8
0;128	Alt-9
0;129	Alt-0
0;130	Alt—
0;131	Alt-=
0;132	Ctrl-PgUp
0;133	F11
0;134	F12
0;135	Shift-F11
0;136	Shift-F12
0;137	Ctrl-F11
0;138	Ctrl-F12
0;139	Alt-F11
0;140	Alt-F12T

Appendix C: List of Tables

TABLE NUMBER	TABLE TITLE	PAGE NUMBER
1	Device Errors	4
2	ANSI Escape Sequences for Controlling the Cursor	10
3	ANSI Escape Sequences to Set the Video Mode	11
4	Parameters for ANSI Video Attribute Escape Sequences	12
5	DOS Command Characters	107
6	DOS Command Keys	108
7	Country Codes for the Country = Command	135
8	DOS Device Names	169
9	Cluster Sizes for Common Types of Disks	181
10	DOS Reserved Area of Various Diskette Types	190
11	DOSShell Shellc Parameters	196
12	DOSShell Command Keys	200
13	DOSShell Special Commands	207
14	Exit Codes for DOS External Programs	241
15	Common File Name Extensions	251
16	Diskette Formats Introduced in Each DOS Release	292
17	Keyboard Nationality Codes	340
18	Linker Switches and Their Effects	348
19	Acceptable Serial Port Parameters	367
20	Display Mode Parameters	369
21	Code Numbers for Alternative Character Sets	371
22	Prompt Meta-strings	399
23	RAMDRIVE.SYS Parameter Values	404
24	Control Characters Not Displayable with the Type Command	479
25	VDISK.SYS Parameter Values	483
A.1	ANSI Escape Sequences for Controlling the Cursor	506
A.2	ANSI Escape Sequences to Set the Video Mode	507
A.3	Parameters for the ANSI Video Attribute Escape Sequences	508
B.1	ASCII Codes	509
B.2	Extended ASCII Codes	515

Appendix D: List of Figures

FIGURE NUMBER	FIGURE TITLE	PAGE NUMBER
1	An ANSI escape sequence	9
2	Menu screen display created by TSR.MNU	15
3	Directory display with Append /x in use	24
4	Display created by the Attrib command	33
5	Directory of a backup diskette	48
6	Source directory of a backup diskette	49
7	A sample BACKUP.LOG file	55
8	USAGE.LOG	82
9	The ChkDsk report	90
10	The DOS 4.0 ChkDsk report	91
11	ChkDsk report for a 5¼" diskette	92
12	Partial output from the ChkDsk /v command	101
13	Sample output from Comp command	115
14	Combining files using wildcard patterns when the target exists	126
15	Combining files using wildcard patterns	127
16	Creating NOBAK.BAT	130
17	The DEBUG Dump command	148
18	Register display in DEBUG	151
19	Writing a file from DEBUG	152
20	Entering a byte at a time	156
21	Filling a section of memory	157
22	Comparing ranges of memory	157
23	Disassembly of FRIDAY.COM	160
24	Loading a boot record in DEBUG	163
25	Displaying the boot record with the Type command	164
26	Displaying the boot record with LIST.COM	165
27	A directory displayed in wide format	171
28	Sample directory tree	174
29	The structure of a directory entry	175
30	A sensible directory tree	179

continues

FIGURE NUMBER	FIGURE TITLE	PAGE NUMBER
31	Disk sector numbering systems	184
32	The DOS reserved areas on a DSDD diskette	191
33	Select's DOSShell parameters screen	195
34	The DOSShell Main Group	199
35	The DOSShell Start Programs menu	200
36	The DOSShell File System menu	202
37	The File System Single File List	203
38	The File System Multiple File List	204
39	The File System System File List	204
40	DOSShell with several submenus added	206
41	The Add Program dialog box	208
42	The Program Parameters dialog box	208
43	Setting up a program	210
44	Fully configured program dialog box	210
45	Output from DATABAK.BAT with Echo statements	222
46	Leaving EDLIN's Insert mode	227
47	Deleting in EDLIN	229
48	Moving lines in EDLIN	232
49	Searching for text in EDLIN	234
50	Queried replacement of text in EDLIN	235
51	Automatic replacement of text in EDLIN	235
52	Inserting text from another file in EDLIN	236
53	Control characters in EDLIN	238
54	Comparing two batch files with FC	257
55	Comparing two batch files with FC, using line numbers	258
56	Comparing program files with FC	258
57	The FDisk Main Menu	263
58	The Partition Information screen	264
59	The DOS 3.3 Create Partition menu	265
60	The Create Partition menu for an empty 32-Mb hard disk	266
61	The attribute byte	273
62	A portion of a directory displayed in DEBUG	277

continues

APPENDIX D: LIST OF FIGURES

FIGURE NUMBER	FIGURE TITLE	PAGE NUMBER
63	Sorting the output of the Find command	281
64	Searching the output of the Sort command	281
65	Output of the Find command	283
66	Output of the Find command with line numbers	284
67	Counting the instances of text using the Find command	284
68	Finding lines not containing the specified text	285
69	Flowchart for ACCOUNTS.BAT	300
70	A screen printed with Graphics loaded	306
71	A graphics screen printed with reversed foreground and background	307
72	A text screen dump with graphics characters	314
73	A text screen with graphics characters dumped as graphics	314
74	A text screen with graphics characters dumped as text	315
75	Hard disk with an interleave factor of 3	324
76	The Mem report	355
77	The Mem/Program report	356
78	PC family memory map	358
79	Codepage status display	374
80	Output from DATABAK.BAT with remarks	419
81	The Select opening screen	434
82	Select's Review Selections screen	437
83	Select's Configuration Parameters screen	438
84	Select's Partition option	439
85	A directory sorted by extension	457
86	An unsorted phone list	458
87	A sorted phone list	458
88	Partial output of the Tree command in DOS 2 and 3	472
89	Partial output of the Tree /f command in DOS 3	473
90	Output of the Tree command in DOS 4.0	474
91	Partial output of the Tree /f command in DOS 4.0	475
92	Output of the Tree /a command in DOS 4.0	476
93	TSR.MNU – Text for screen display	480
94	The effect of the question-mark character	493

Appendix E: List of Programs

PROGRAM TITLE	PAGE NUMBER
TSR.BAT: A TSR Menu Program	15
AUTOEXEC.BAT File for dBASE III PLUS with Diskettes	37
AUTOEXEC.BAT for WordPerfect with Diskettes	38
AUTOEXEC Code for a Diskette and a RAM Disk	39
AUTOEXEC Code to Place COMMAND.COM on a RAM Disk Using DOS 2	40
AUTOEXEC.BAT for a Hard-Disk–Based System	43
DATABAK.BAT	60
BUSLTR.BAT	64
STYLE.BAT	66
USEAUTO.BAT	67
NOAUTO.BAT	67
KEY.BAT	78
LOGON.BAT	80
A CONFIG.SYS File	119
A CONFIG.SYS File for Use with DESQview	120
CONFIG.SYS File for a Quick Start	121
TAKEOVER.BAT	139
DISMISS.BAT	139
Part of HDBKUP.BAT, Showing Ctty Commands	140
DATABAK.BAT, with Messages in Echo Statements	221
CHKCOPY.BAT	244
FRIDAY.DEB	245
TESTFRI.BAT	246
TESTFRI.BAT, Version 2	246
CLOCK.BAT	249
ACCOUNTS.BAT	298
UPDATE.BAT	318
DIRBAK.BAT	319
HDBACKUP.BAT	320
AUTOEXEC.BAT code for Belgium	378

continues

APPENDIX E: LIST OF PROGRAMS

PROGRAM TITLE	PAGE NUMBER
MFORMAT.BAT, version 1	413
MFORMAT.BAT, version 2	414
SETDATE.BAT	415
BUSLTR.BAT with Remarks	418
DATABAK.BAT with Remarks	420
Sample AUTOEXEC.BAT File Created by Select	431
Sample AUTOEXEC.BAT /DOS 3.3 Select	432
MCOPY.BAT	452
MCOPY.BAT, Version 2, with an Environment Variable	453
RAMAPP.BAT	485
RWS.BAT	486
DATEBAK.BAT	497

General Index

" (double quote), 107
% (percent), 59. *See also* Replaceable parameters in batch files
* (asterisk), 107
+ (plus sign), 107
, (comma), 107
- (hyphen), 464
. (period), 107, 173, 387
.. (double period), 84, 173, 386
/ (slash), 107, 464
: (colon), 107
; (semicolon), 107
< (less than), 107, 285, 343, 411
= (equal to), 107
= = (double equal sign), 329
> (greater than), 107, 195, 343, 411
> > (double greater than), 411
? (question mark), 107, 491–92, *493*
@ (at sign), 350
[(left bracket), 195
\ (backslash), 107, 343, 384, 385
\BACKUP directory, 47
\BATCH directory, 63, 178, 382
\DOS directory, 178
\HDBACKUP directory, 322
\UTILITY directory, 178, 382
¦ (double bar), 107, 209, 411
^ (caret), 25–26
^@ (ASCII 0), 222
~ (tilde), 14
Abort, Retry, Ignore, Fail? prompt, 3–6
Absolute path name, 384–85
ACCOUNT.BAT file, 333
  commands, *298–99*
  flowchart, *300*
Allocation unit, 294. *See also* Disk cluster(s)
Allocation unit report, DOS 4.0 ChkDsk command, 91
American National Standards Institute (ANSI), 6
American Standard Code for Information Interchange (ASCII). *See* ASCII code; ASCII code, extended; ASCII files; ASCII string
ANSI.SYS file, **6–19**
  escape sequences
    cursor movement, screen color, and prompt color determined by Prompt command and, 401–2
    displaying a clock on-screen with, 249
    listing, 506–8 (tables)
    procedure, 7–13
    sending to printers, 26–27
  examples, 13–17
    enhancing text display, 13–14
    menus and help screens, 14–17
    menu screen display created by TSR.MNU, 15–17
    swapping key positions, 14
  overview, 6–7
  tips, 18–19
  warnings, 17–18
Append command, **19–25**, 111
  examples, 23–24
  overview, 19–20
  procedure, 20–23
    extended scope of Append, 22
    extending searches in DOS 4.0, 22–23
    placing append lists in the environment, 21–22
    standard command, 21
  tips, 25
  TSRs invoked by, 476
  warnings, 24–25
Append command, EDLIN, 236
APPEND.COM program, 20
Applications programs with AUTOEXEC.BAT files on distribution diskettes, 45
Archive file attributes, 33–34, 275
ASCII code, **25–27**, 145, 302, 311, 509–15(table)
  key redefinition and, 13
ASCII code, extended, 13, **27–28**, 515–16(table)
ASCII files, 57
  sorting, 454–56
ASCII strings, 238, 329
.ASM files. *See* Assembly language (.ASM) programs
Assemble command, 159
Assembly language (.ASM) programs
  creating with Assemble, 159
  debugging commands for, 161–62
  examining with Unassemble, 159–60
  turning assembled code into executable programs, 347–50
Assign command, **28–31**
  Join and, 30, 335
ASSIGN.COM program, 29
Attrib command, **31–35**, 274, 275
  addressing the archive attribute, 33–34
  addressing the read-only attribute, 32
  displaying file attributes, 32, *33*
  repeating, on subdirectories, 34
AutoCAD (software), 42
AUTOEXEC.400 file, 435–36
AUTOEXEC.BAT file, **35–45**
  Attrib command with, 34
  code to nationalize the computer for Belgium, 378(table)
  on diskette-based systems, 36–38

on diskette-based systems with a RAM disk, 38–41
DOSShell added to, 195
on hard-disk based system, 41–44
Install = as alternative way of loading TSRs, 333–34
overview and procedures, 35–36
sample, created by DOS 3.0 and 3.1 Select command, 430, *431*
sample, created by DOS 3.3 Select command, *432–33*
sample, created by DOS 4.0 Select command, 435–436
special hardware considerations, 44–45
switching among different, 66–67
warnings and tips on, 45–46
AUX, 169, 290

Backup
full, 47, 50–51
hard disk, 53, 56, 79, 139–41, 269, 270, **315–23**
current data, 317–18
ensuring timely, 320–23
large directories, 319–20
new software, 317
incremental, 33, 47, 51
partial, 47
Backup command, **46–56**, 316
archive attributes and, 33–34, 275
effect of Assign on, 30
effect of Join on, 337
exit codes, 240, 241(table)
overview, 46–49
command characteristics, 47–49
types of backups, 46–47
procedure/examples, 49–53
full, incremental, and partial, 50–52
selecting files by date and time, 52–53
restoring files backed up by, 426–29
tips, 53–56
formatting as you go, 53
handling incompletely filled backup diskettes, 56
keeping records of backup files, 53–55
warnings, 53
Backup files
avoiding making backups of, 34
importance of making, 99
keeping record of, 53–54, *55*
BACKUP.LOG file, 54, *55*
.BAK files. *See* Backup; Backup files
.BAS BASIC program files, 251
Batch file(s) (.BAT), **56–69**
adding remarks to, 417–20
blank screen lines in, with Echo, 221–22
creating with Copy CON, 130
debugging, and Echo command, 223
ErrorLevel modifier for If command used in, 240–46
examples, 59–67
creating alternative command names, 61
creating convenience commands, 62
executing fixed sequence of commands repeatedly, 59–61
repeating commands on different files, 61–62
saving and restoring defaults, 65–66
simplifying use of hard disk, 62–65
switching among configurations, 66–67
executing secondary, 104
languages for extending capabilities of, 302
overview, 56–57
procedure, 57–59
batch programming instructions, 58–59
creating/invoking, 57–58
executing batch files from another, 58
passing parameters to, 59
processing order, 111
testing environmental variables in, with Set, 441–42
tips, 68–69
using as subroutines, with Call command, 77–82
warnings, 67–68
Baud rate, 366
Belgian keyboard, 339, 378(table)
Binary file(s) (.BIN), comparing, 257–58. *See also* Command file(s) (.COM)
BIOS (Basic Input/Output System), 357
ROM, 183, 184
Bit bucket. *See* NUL device
Bit-mapped graphics mode images, 304
Bits formatted by Mode for data transmission to serial ports, 365–67
Blinking video, 13
Boot record, **69–71**, 189
hard disk, 327
loading, displaying in DEBUG, 163–65
Break command, **71–73**, 117. *See also* Ctrl-Break
checking current break status, 72
as configuration command, 72
as DOS command, 72
warning and tips on, 72–73
Buffer
creating continuous space, with FastOpen, 252–54
creating disk, with Buffers =, 73–77
Buffers = command, **73–77**, 111, 117, 253, 501
BUSLTR.BAT file, *64*
showing remarks, *418*, 420

Byte(s), 365
Byte editor, 145–46

Call command, 58, **77–82**, 301
  examples, 78–81
    creating batch subroutines, 78–79
    creating routines for special circumstances, 79
    usage-logging program LOGON.BAT, 79–81, 82
  overview/procedure, 77–78
Case matching of characters, 330–31
CATALOG.HD file, 34
CD (change directory) command, **82–85**, 386, 406
  changing directories on noncurrent drives, 85
  finding out which directory is current, 85
  making a directory current with, 83–84
  moving up one level in the directory, 84–85
Cell(s), text mode characters displayed in, 304
Characters
  ASCII (*see* ASCII code)
  case matching, 330–31
  command, **107**
  control, 25, 237–38, 311
  space, 26
  wildcard (*see* Wildcard characters)
Character sets, **370–72**. *See also* Foreign language characters; Graphics characters
  checking the status of, 374
  code page numbers for alternative, 371(table)
  preparing alternative, for the printer, 372–73
  using alternative, 373–74
ChCP (change code page) command, **86–87**, 137
ChDir. *See* CD (change directory) command
CHKCOPY.BAT, 243, *244*, 245–46
CHKDSK.COM/.EXE file, 39, 88
ChkDsk command, **88–102**, 272, 274
  damage not repairable by, 98–99
  effect of Assign on, 30
  effect of Join on, 337
  overview, 88–93
    disk reports, *90*, *91*, 92–93
  procedure
    checking file fragmentation, 95–99
    eliminating lost clusters, 94–95
    getting a disk report, 93
  reports of, vs. Mem command report, 355
  vs. Tree command, 471, 474
Child directories, 173
Clock, displaying on the screen, 249
CLOCK.BAT file, *249*
CLS (clear screen) command, **102**
  clear-screen escape sequence as a substitute for, 18
CMOS chip, 142, 444, 468
Code page information file, 372, 373
Code page numbers, 302, 303
  for alternative character sets, 370, 371(table)
  change, 86–87
  status display, *374*
Color
  graphics screens in, 307–8
  shell parameters for, 198
Color monitor, 45
.COM files, 247. *See also* COMMAND.COM
  processing order, 111
Command(s)
  batch programming language, 58
  for changing CONFIG.SYS default values, 117
  controlling display of, with Echo, 218–19
  creating alternatives or shortening names for, 61
  creating convenience, 62
  DEBUG, 146–47, 161–62
  DOSShell special, 207(table)
  EDLIN, general rules for, 224–25
  extended-memory, 162
  path names used in, 386
  processing order, **111**
  processing order, CONFIG.SYS file, 118
  program debugging, 161–62
  receiving from non-console points, 138–41
  repeating, on different files, 61–62
  repeating a group of, with GoTo, 295–96
  repeating execution of a fixed sequence of, 59–61
  suppressing display of, in DOS 3.3 and later, 219
  syntax, xiii (*see also individual commands*)
  versions, xi (*see also* DOS versions)
  which invoke TSRs, 476–77
Command characters, **107**, 398
COMMAND.COM, 35, 81, 465. *See also* Environment space
  AUTOEXEC code to place, on a RAM Disk, *40*
  configuring with Shell=, 448–49
  loading secondary, 103, 104, 448
  removing secondary, with Exit, 248–50
  replacing the original, 105
Command command, 58, **103–6**
  executing a secondary batch file with, 104
  loading command processor with, 103, 104

replacing the original command processor with, 105
setting the environment size with, 105–6
Command file(s) (.COM). *See also* Binary file(s) (.BIN)
converting from .EXE to, 247–48
Command keys, **108–10**, 237
DOSShell, 200(table)
keys and effects, 108–9(table)
procedure/examples, 110
Command processor. *See* COMMAND.COM
Communications ports. *See* Serial port(s)
Com*n* name for serial ports, 112, 169
Comp command, **112–16**, 259
examples, 114–15
overview/procedure, 112–14
sample output, *115*
tips, 115–16
CON device name, **116**, 169, 411
copy, 129–31
CONFIG.400 file, 435
CONFIG.SYS file(s), **116–22**
adding remarks to, 417–20
Attrib command with, 34–35
changing character sets and, 371–72
configuration commands, 72, 134, 167–68, 217, 259, 333, 346, 447, 459, 463
creating, 117–18
determining number of file handles in, 280
examples, *119–20*
for use with DESQview, *120*, 121
for use with Quick Start, *121*
installing TSRs in, 45–46, 252
loading device drivers in, 7, 116, 168, 213
order of commands, 118
overview, 116–17
search paths and files loaded by, 383
switching among, 66–67
warnings and tips on, 122
Console, 116, 411
Continuous space buffers, creating with FastOpen, 252–54
Control characters, 25, 311
EDLIN handling of, 237, *238*
not displayed with Type command, 478, 479(table)
Control program, 361
Conventional memory, 357, *358*
Copy. *See* Copy command; DiskCopy command; Replace command; Xcopy command
Copy35 and Rcv35 commands, **131–33**
COPY35.COM file, 131
Copy command, 33, 47, 60, 95, **122–29**, 186, 216, 275, 301, 489
overview, 123
procedure/examples, 123–27
combining files, 124–27
copying files, 123–24
redirecting files, 127
tips, 128–29
warnings, 127–28
Copy CON command, 57, **129–31**
Country = command, 117, **133–36**, 137
COUNTRY.SYS file, 134, **136–38**
Cross-linked files, 98
Ctrl-Break, 68. *See also* Break command
Ctrl-C, 68
Ctrl-G, 26
Ctrl-I, 26
Ctrl-M, 26
Ctrl-P, 54
Ctrl-PrtSc, 54
Ctty (change teletype) command, **138–41**
HDBKUP.BAT showing, *140–41*
CURRENT.BAT file, 322, 416
Cursor, ANSI escape sequences for controlling, 9, 10(table), 506(table)
Cylinders, hard-disk, 326

DAILYBAK.BAT file, 322
Data
backing up current hard disk, 317–18
in DEBUG, 145–46
entering new values, 155–56
fill bytes with, 156
searching for, 153–55
erasing, in DOS partitions, 269
filtering, 280–81
format of, created by Mode for data transmission, 365–66
DATABAK.BAT file, 60, 496, *497*
with messages in echo statements, *221*, *222*
output with remarks, *419*, *420*
Data bits, 365
Data Migration Facility, 132
Date and time, 36, 176. *See also* Time command
displaying/changing date, 141–44
placing on environment with SET command, 143, 415
selecting files by, 52–53, 496–98
Date and time stamp
changing, on files, 129
on disks, 99–101
DATEBAK.BAT file, 246
Date command, **141–44**, 415
dBASE III PLUS, AUTOEXEC.BAT file for, 36, *37*
DBASE.DTX file, 37
Dead keys, 340
DEBUG.COM/.EXE, 147
DEBUG utility program, 26, 108, **144–66**
altering file attributes with, 275–77
examples, 162–65
handling corrupted boot records with, 70
low-level formats with, 325
overview, 144–47
command rules, 146–47

data in DEBUG, 145–46
memory addresses, 146
procedure, 147–65
comparing sections of memory, 156–57
creating a program, 158–59
doing hexadecimal arithmetic, 158
entering new values, 155–56
examining/changing register contents, 151
examining memory, 148–49
examining program instructions, 159–60
exiting DEBUG, 147
extended-memory commands, 162
filling a section of memory, 156
invoking DEBUG, 147
loading disk sectors, 150
loading programs, 149–50
moving data in memory, 157–58
program debugging commands, 161–62
searching for data, 153–55
writing files to disk, 151–52
writing to disk sectors, 152–53
script files
input program FRIDAY.COM, 244–46
redirecting input with, 413
warnings and tips, 165–66
Defaults, saving and restoring with batch files, 65–66
Del (delete) command, **166–67**
Delete command, EDLIN, 229–30
DESQview (software), 337
CONFIG.SYS file for, *120*, 121
Device = command, 117, **167–68**, 360. *See also* DRIVER.SYS file
Device driver(s)
ANSI.SYS compatible (*see* ANSI.SYS file)
in CONFIG.SYS file, 116–17
DISPLAY.SYS, 192–93
DRIVER.SYS, 213–17
loading with Device = , 167–68
PRINTER.SYS, 396–97
RAMDRIVE.SYS, 402–4
VDISK.SYS, 481–87
XMA2EMS.SYS, 500–503
XMAEM.SYS, 499–500
Device errors
Abort, Retry, Ignore, Fail? prompt for, 3–4
error messages, 4–6(table)
Device names, **169**
Com*n*, 112
CON, 116
LPT*n*, 351
NUL, 379
PRN, 398
PRT, 398
Device Status Report escape sequence, 13
DIRBAK.BAT file, *319*
Dir command, 167, **170–71**
use of wildcards with, 490, 493
Directories, *172–80*. *See also* Subdirectories
backing up large, 319–20
of backup diskette, *48*
changing, on noncurrent drives, 85
contents of directory entries, 175
copying complete, with Xcopy, 493–96
creating virtual drives by assigning new drive name to name of an existing, 460–63
creating with MD command, 351–54
current directory as a path, 385–86
danger of copying files to another, 127–28
determining current, 85
displaying complete, 171–72
displaying names of, 180
displaying partial, 171
executing files on noncurrent, 64
information in, 172
invoking files on nested, 65
listing of, with Tree command, 470–76
making current, 83–84
moving up one level in, 84–85
naming subdirectories, 179–80
nesting subdirectories, 178–79
organizing trees, 176–78
parent and child, 173
path names (*see* Path names)
remove, 405–7
root (*see* Root directory)
sorting current directory by extension, 457
source directory of backup diskette, *49*
tree-structured, 172–73, *174*
Disk. *See also* Diskette system; Hard-disk system; RAM disk
assigning internal volume label to, with Label, 344–46
boot record, 69–71
bringing text to memory from, 236
checking with ChkDsk command, 88–102
date and time stamps on, 99–101
formatting, 53, 288–94, 327
location of files on (*see* File allocation table (FAT))
recovering disks with a bad FAT, 409–10
writing text from memory to, with EDLIN, 235–36
Disk buffers, 73–77
Disk-caching program, 77
Disk cluster(s), 91, **181–83**, 271, 272, 294
different files occupying the same, 98
eliminating lost, 94–95

location on hard disk, 327
sizes of, for common disk types, 181(table)
warnings and tips, 182–83
DiskComp command, **184–86**
effect of Join on, 337
exit codes, 240, 241(table)
DISKCOPY.COM file, 39
DiskCopy command, 60, 61, **187–89**, 216
vs. Copy command, 128
exit codes, 240, 242(table)
Disk drive(s)
assigning new letter designation to, 28–31
changing directories on noncurrent, 85
determining maximum number of virtual, with LastDrive =, 346–47
treating disk drives as subdirectories of other, with Join, 335–37
Disk error messages, 4–6(table)
Diskettes, structure of, **189–91**
DOS reserved areas, 190(table)
DOS reserved areas, DSDD diskette, *191*
Diskette system. *See also* Disk
AUTOEXEC.BAT file for, 36–38
with RAM disk, 38–41
backups on, 48, 49
incompletely filled diskettes, 56
comparing diskettes with DiskComp, 184–86
copying diskettes with DiskCopy, 187–89
disk cluster size on common types of, 181(table)
error messages, 4–6(table)
formatting diskettes, 290–92
assigning a name, 290–91
formats introduced in each DOS release, 292 (table)
to less than capacity, 291–92
system, 290
installing an external diskette drive, 215
installing a second logical drive in an existing physical drive, 215–17
Disk optimizing software, 97
Disk report, 90–93
getting, 93
Disk sectors, 181, **183–84**, 189
hard disk, 326
interleave factor, 323–25
loading in DEBUG, 150
relationship of DOS and BIOS, *184*
writing files to, in DEBUG, 152–53
DISMISS.BAT, *139*
Displaying graphics characters, 312–13
Display modes, 369–70
parameters, 369(table)
DISPLAY.SYS file, **192–93**
DOS command(s)
Break as, 72
keys, 108–10
suppressing display of, in DOS 3.3, 219
DOS reserved area on diskettes, 190(table), *191*
DOSShell program, 20, **194–213**
command keys, 200(table)
configuring the shell, 195–98
creating alternative shells, 211–12
creating a menu system, 205–11
adding submenus, 205, *206*
Add program dialog box, 206, *208*
program dialog box, 209, *210*
Program Parameters dialog box, *208*, 209
setting up a program, 209, *210*
customizing, 195
file system, 201–5
menu, *202*
Multiple File List, *204*
Single File List, *203*
System File List, *204*
Main Group options, 201
overview, 194–95
shellc parameters, 196–97 (table)
shell use, 198–99
special commands, 207(table)
Start Programs menu, 199, *200*, 201
TSRs invoked by, 476
warnings and tips, 212–13
DOS versions, xiv
determining, with Ver command, 487
diskette formats introduced in each, 292(table)
DOS 2.x
ASCII sorts in, 454–56
hard disk formatting in, 293
path extension utilities, 63
placing COMMAND.COM on a RAM disk with AUTOEXEC code using, *40*, 41
Tree command, 471, *472*
DOS 3.0
hard disk formatting in, 293
Link command, 349
Select command, 430–31
sorts in, 454–56
Tree command, 471, *472*
DOS 3.1
configuring COMMAND.COM with Shell =, 448–49
Join and Subst commands, 30
Select command, 430–31
DOS 3.2
Backup command, 54
configuring COMMAND.COM with Shell =, 449

Select command, 431–33
DOS 3.3
Append command, 21, 24–25
Backup command, 54
disk buffers, 75–76
Fail option to device error prompt, 3
increasing search speeds in, 111
Select command, 431–33
suppressing command display in, 219
DOS 4.0
ANSI.SYS, 7, 13
Append, 20, 21, 22–23, 111
boot record, 71
buffers, 254
ChkDsk, 90, 91
diskette formatting, 292
DOSSHELL, 194
expanded memory in, 500–503
Graphics command, 305, 308
handling serial port errors, 368
hard disk formatting, 293–94
hard disk structuring in, 261–62, 267–68
installing, without using Select command, 438–40
installing TSRs in, 45
Select command, 433–36
setting number of lines on the screen, 370
setting parity bits in, 366
Shell = special features, 449
Tree command, 471, *474*, *475*, *476*
Downward-compatibility, Format command, 291
Driveparm = command, 117, **217**
DRIVER.SYS file, 119, 120, 188, 213–17
installing external diskette drive, 215
installing external hard drive, 215
installing second logical drive, 215–17
selecting switches and values, 213–15
Dump command, DEBUG, *148*-49, 154, 277

Echo command, 58, **218–23**
appending files with, 220
controlling display of commands with, 218–19
creating blank screen lines with, 221–22
debugging batch files with, 223
displaying user messages with, 219–20
example of, 220–21
effect of, on Pause command, 388–89
passing escape sequences with, 8
sending characters via a pipe with, 414
turning off program messages with, 221
turning off the prompt with, 223
Editing keys, 108–9(table)
EDLIN utility program, 57, 108, **223–38**
overview, 223–25
command rules, 224–25
line logic, 223–24
memory limitations, 224
screen, 225
procedure, 225–37
adding text from another file, 235
bringing text from disk to memory, 236
copying lines, 231–32
creating new files, 225–26
deleting lines, 229–30
editing existing files, 226
editing existing lines, 230–31
ending the session, 236–37
inserting lines, 226–27
moving lines, 232
replacing text, 234–35
searching for text, 233–34
viewing files, 227–29
writing text from memory to disk, 235–36
tips on handling control characters, 237–38
EMS Allocate command in Debug, 162
EMS Deallocate command in Debug, 162
EMS Map command in Debug, 162
EMS Status command in Debug, 162
END command, EDLIN, 237
End-of-file marker, 124–25
Enhanced expanded memory standard (EEMS), 361–62
Enter command, DEBUG use of, 156
Environmental variable(s)
Path as, 381
testing for, 329–30
testing of, in batch files with Set, 441–42
view current, with Set, 239
Environment space, 103, **238–40**
enlarging, 239–40
joins and, 337
placing variables in, with Set, 440–43
setting the size of, 105–6
variables (*see* Environmental variable(s))
viewing, with Set, 442
equivalence, testing for, 329–30
Erase. *See* Del (delete) command
ErrorLevel modifier, If command, 56, **240–46**
examples, 243–46

exit codes tested by, 241–43(table), 304, 340, 429
overview, 240–41
procedure, 241
Error messages
Abort, Retry, Ignore, Fail? prompt, 3–4
device errors, 4–6(table)
Escape sequences, ANSI.SYS, 8, *9*
cursor control with, 9, 10(table), 506(table)
displaying time on the screen with, 249
key redefinition with, 12–13
miscellaneous, 13
video attribute set with, 12(table), 508(table)
video mode set with, 10, 11(table), 507(table)
Even parity, 365–66
EXE2BIN, **247–48**
Executable (.EXE) files
cannot be directly examined in DEBUG, 147
converting, to command files, 247–48
processing order, 111
search for (*see* Path command)
turning assembled code into, 347–50
Execution, temporary interrupt of, using Pause command, 387–89
Exist modifier, If command, 128, 331
Exit codes, 240–46
DOS external program, 241–43(table)
testing for
with ErrorLevel, 240–42
with If command, 332
Exit command, 105, **248–50**
Expanded memory, 359–62
XMA2EMS.SYS device driver to manage, 500–503
Extended Batch Language (EBL) (software), 302
Extended memory, 359, 361–62
commands, 162
XMAEM.SYS device driver to manage, 499–500
Extensions, filename, 250, 251(table)
sorting by, 457

FastOpen command, 75, 111, **252–54**, 501
TSRs invoked by, 476
FASTOPEN.EXE file, 253, 333
FCBs = command, 117, **259–60**, 446–47
FC command, **255–59**
comparing binary files, 257, *258*
comparing text files, 256, *258*
comparing text files using line numbers, 257, *258*
switches used with, 255(table)
warnings and switches on, 259
FDISK.COM file, 262, 270
FDisk command, **261–70**
effect of Join on, 337
overview, 261–62
procedure, 262–69
changing the active DOS partition, 269
creating a DOS partition, 264–68, 327
deleting the DOS partition, 269
displaying DOS partition information, 264
warnings and tips, 270
File(s). *See also* Batch file(s) (.BAT); Program file(s); Text file(s)
adding and removing, from print queues, 393–94
appending, 220, 412, 413–14
attributes of normal, 273 (*see also* File attribute(s))
avoiding overwriting existing, 128
combining with Copy, 124–27
comparing with Comp, 112–16
controlling access to, with Share, 445–47
copying with Copy, 123–24
creating with EDLIN, 225–26
deleting, 166–67
editing an existing, with EDLIN, 226
editing lines in, 226–27, 229–32
finding instances of text in, with Find filter, 284
finding text in, with Find filter, 283, *284*
hidden, 92–93, 407
information on, in directory entries, 172
inserting lines in, with EDLIN, 226-27
keeping a record of backup, 53–55
line logic in, 223–24
listing with ChkDsk, 101, *102*
location of, on disk (*see* File allocation table (FAT))
recovering, with bad sectors, 408–9
redirecting, 127
rename with Ren, 421–22
repeating commands on different, 61–62
searching for or editing text in, with EDLIN, 233–36
search path for nonexecutable, 19, 20
selecting by date, 52–53, 496–98
selecting by name, 496
testing for the presence of a, 331
transferring, from PC to PS/2 parallel ports, 131
viewing, with EDLIN, 227–29
writing to disk in DEBUG, 151-52
File allocation table (FAT), 176, 189, 216–17, **270–72**
danger of overwriting disk FAT, 3
errors in reported by ChkDsk, 97–98
information contained in addresses of, 271
location on hard disk, 327
recovering disks with bad, 409–10

File/Associate command in DOSShell, 195, 212
File attribute(s), **273–78**
  addressing the archive, 33–34
  addressing the read-only, 32
  altering, with DEBUG, 275–77
  attribute byte, 32, 175, *273*
  displaying, 32, *33*
  examining/changing with Attrib, 31
  overview, 273–75
    archive files, 275
    hidden files, 274
    normal files, 273
    read-only files, 273–74
    subdirectories, 275
    system files, 274
    volume labels, 274
  procedure, 275
  warnings, 277–78
File comparison utility, 112–116, 255–59
File control blocks (FCB), 259–60, 280, 446–47
  error messages, 6(table)
File fragmentation, 189, 252, 410
  checking for, 95–99
  eliminating on hard disks, 272
File handles, 280
File names, **278–79**
  changing, 421–22
  extensions for, 250, 251(table)
  rules for, 278
  tips on, 278–79
Files = command, 117, **279–80**
File System, DOSShell, 201–5
  menu, *202*
  Multiple File List, 203, *204*
  Single File List, 202, *203*
  System File List, 203, *204*
Filter(s), 280–81, 411. *See also* Find filter program; More filter program; Pipes; Redirection; Sort filter program
FIND.EXE file, 282
Find filter program, 100, 281, **282–85**
  overview, 282
  procedure, 282–85
  counting instances of text in a file with, *284*
  counting lines not containing the text with, 284, *285*
  finding text in a file with, *283*, *284*
  using in a pipe, 285
Flow of control, and GoTo command, 298–301
Flow of execution, redirecting with If, 332
Foreign language characters, 26, 86–87, 192–93, 311. *See also* Character sets; Mode command
For...In...Do command, 58, **286–88**, 317
  examples, 287–88
  overview/procedure, 286–87
  tips, 288
  used instead of Replace, 425
FORMAT.COM/.EXE file, 39, 53, 289
Format command, 90, 268, **288–94**, 489
  effect of Join on, 337
  exit codes, 240, 242(table)
  overview, 288–89
  procedure/examples, 289–94
    diskettes, 290–92
    giving disks a name, 290–91, 345
    hard disks, 293–94, 324, 327
  for use with DOS 3.3 and Select, 431–32
  warnings and tips, 294
Formatting of disks when backing up, 53
Framework (software), 313
FRIDAY.COM, 244–46, 322
  disassembly of, *160*
FULLBACK.BAT file, 322
Full backup, 47, 50–51
Function key(s), 27

Game ports, 389, **390**
GETENV C function, 240
Go command, 161
GoTo command, 58, **295–302**, 420
  conditional execution and subroutines with, 296–97
  directing flow of control with, 298–301
  overview, 295
  repeating a group of commands with, 295–96
  tips, 301–2
GrafTabl command, **302–4**, 371
  exit codes, 240, 242(table)
  TSRs invoked by, 477
GRAPHICS.PRO file, 308
Graphics board, multifunction, 45
Graphics characters, 26, 302–4, **311–15**
  displaying, 312–13
  entering, 311–12
  high-order vs. control, 311
  printing, 313, *314*, *315*
GRAPHICS.COM file, 305
Graphics command, **304–11**
  overview, 304–5
  procedures for printing graphics screens, 305–11
    in color, 307–8
    editing the Profile File, 309–11
    from the IBM Convertible, 309
    in inverse mode, 306–7
    as a wide carriage, 308
  TSRs invoked by, 477
Graphics mode, 304–5
  printing graphics characters, 313–15
  printing screens in, 306–11
Graphics tables, loading with GrafTabl, 302–4

Hard-disk system. *See also* Disk
  AUTOEXEC.BAT file on, 41–44
  back up and restore, 53, 56, 79, 139–41, 269, **315–23**
    examples, 317–23
    four alternative procedures for, 316–17
    overview, 315–16

disk clusters, 271
  chains of, 272
  sizes for common types of, 181(table)
  upgrading 10-megabyte hard disks and, 182
eliminating file fragmentation on, 272
formatting hard disks, 293–94, 327
  low-level, 324–25
hard disks structure, **326–28**
installing an external hard drive, 215
interleave factor, **323–25**
  overview, 323–24
  procedure for changing, 324–25
navigating in, with batch files, 62–65
preparation with FDisk command, 261–70
  changing active partitions/deleting partitions, 269
  creating partitions, 264–68, 327
  displaying partition information, 264
Hardware
  program to run special, located on AUTOEXEC.BAT, 44–45
  setup, 443–45
Hardware interrupts, processing with stack frames, 459–60
HDBACKUP.BAT file, 79, 245, *320–22*, 323, 416
HDBKUP.BAT file, Ctty commands in, *140–41*
Header, file, 149
Help screens, using ANSI.SYS to create complex, 14–17
Hercules Graphics Card, 45
Hexadecimal arithmetic, 158
Hexadecimal data, 145, 149
HEX command, 158
.HEX files, 149
Hidden files, 92–93, 407
  attributes, 274
High-order graphics characters, 313
  displaying with Type, 312–13
Hot key, 477

IBMBIO.COM, 70, 190, 274
IBM Convertible computer, printing graphics screens from, 309
IBMDOS.COM, 70, 190, 274
IF command, 58, 296, **328–33**
  ErrorLevel modifier (*see* ErrorLevel modifier, If command)
  examples, 323–33
    nesting Ifs, 332–33
    testing a parameter, 332
  overview, 328–29
  procedure, 329–32
    case matching, 330–31
    testing for equivalence, 329–30
    testing for exit codes, 332
    testing for the presence of a file, 331
  tips, 333
If ErrorLevel command, 58
If Exist command, 58
If Not command, 58
Incremental backup, 33, 47, 51
Input, redirecting, 412, 413
Input (program debugging) command, 161
Insert mode, EDLIN, 226–27
Install = command, 20, 117, 252, **333–34**
Interleave factor, hard disk, **323–25**
  overview, 323, *324*
  procedure for changing, 324–25
International telephone prefixes, 135(table)
Inverse video, 13
  graphics mode, 306–7
IO.SYS, 70, 190, 274

Join command, **335–37**
  canceling a join, 336
  commands which become unreliable when using, 30, 337
  examples, 337
  joining drives to main drive, 336
  overview, 335
  viewing joins in effect, 336
  warnings and tips, 337
JOIN.EXE, 335

Key(s). *See also* ASCII code; ASCII code, extended
  ANSI escape sequences for redefining, 12–13, 14, 17–18
  dead, 340
  DOS command, 108–10
  hot, 477
KEY.BAT, 78
KEYB.COM file, 333, 338
Keyb command, **338–40**
  exit codes, 240, 242(table)
Keyboard(s)
  country codes for, 136
  extended functions (*see* ANSI.SYS file)
  nationality codes, 340(table)
  remapping to different national, 338–43
KEYBOARD.SYS file, 338–339, **341–42**
Keyb*xx* command, *342–43*

LABEL.COM/.EXE file, 344
Label command, 90, 274, **344–46**
LASTBKUP.BAT file, 322, 323
Lastdrive = command, 117, **346–47**, 460
LIM-EMS standards, 354, 360
Line(s)
  comparing text files using, 257, *258*
  counting, not containing the text with Find filter, 284, *285*
  editing in EDLIN
    copying, 231–32
    deleting, 229–30

editing existing, 230–31
inserting, 226–27
line logic, 223–24
moving, 232
setting number of, on the screen, 370
sorting, in text files, 454–58
Link command, **347–50**
overview, 347–48
switches, 348(table)
usage, 348–50
LINK.EXE file, 348
LIST.COM, displaying boot record with, 164, *165*
List command, EDLIN, 227–29
Load command, 149
Logical drive. *See* virtual drives
LOGON.BAT file, *79*, *80–81*, 332, 402, 416
Loops
exit conditions for, 296
For...In...Do command, 286–88
GoTo, 295–96
using Shift commands in, 451–52
Lotus 1-2-3 (software), 23, 59
Lotus-Intel-Microsoft Expanded Memory Standard, 354
LPT*n* line printer device name, 169, **351**

Macros, 485
Mark parity, 366
MCOPY.BAT file, 332, *452*
with environmental variable, *453*
MD (Make Directory) command, **351–54**, 386
MEM704.COM file, 359
MEM.COM file, 354
Mem command, **354–56**
Mem/Program report, *356*
Mem report, *355*
Memory, **357–62**. *See also* Environment space
adding, 359
comparing sections of, in DEBUG, 156-*57*
conventional, 357
bringing text from disk to, 236
determining free, with ChkDsk, 88, 93
examining, in DEBUG, 148–49
expanded and extended, 359–62
commands for handling extended, 162
filling a section of, in DEBUG, 156, *157*
increasing free, 100, 102
limitations on, and EDLIN, 224
PC family memory map, *358*
read-only, 357
speed, 362
summary report created with Mem command, 354–56
system, 359
writing text to disk from, with EDLIN, 235–36
Memory addressing, 357–59
DEBUG use of, 146, 148, 150, 153
Memory boards and cards, 77, 254, 360
Memory bus, 362
Memory pages, 501
Memory segments, 357
Menu(s)
ANSI.SYS escape sequences for creating complex, 14–17
building system of, with DOSShell, 205–11
Message(s). *See also* Error messages
displaying user-created, with Echo, 219, 220–21
turning off, from programs, 221
MFORMAT.BAT, *413*, *414*
Microsoft Windows (software), 337, 404
Microsoft Word (software), 313
MkDir command. *See* MD (Make Directory) command
MODE.COM/.EXE file, 364
Mode command, **363–75**
alternative character sets, 370–72
checking the status of, 374
preparing, for the printer, 372–73
using, 373–74
ANSI.SYS /l switch and, 8
character set codes, 340
configuring the printer, 368–69
configuring the screen, 369–70
configuring the serial port, 364–67
escape sequences used with, 10, 11(table)
output to serial printer, 44–45
overview, 363–64
redirecting printer output, 367–68
tips on, 375
TSRs invoked by, 477
Modems, 389, 390
Monitors, 45
switching between color and monochrome, 375
MORE.COM file, 375
More filter program, 281, **375–77**, 413, 415
MS-DOS disk buffers, 74
MSDOS.SYS file, 70, 190, 274

Name command in Debug, 149
National language(s)
character sets, 86–87, 192–93, 311, 370–75, 377
information on (*see* COUNTRY.SYS file)
keyboards, 338–43
Nested commands, If, 332–33
Network(s)
controlling access to files on, 445–47
error message on, 6(table)
limiting access by creating alternative shells, 211–12
NLSFunc (national language support function), 371, **377–79**
AUTOEXEC.BAT code for Belgium example, 378(table)

TSRs invoked by, 477
NLSFUNC.EXE file, 334, 377
NOBAK.BAT, creating with Copy Con, 130
Nonexecutables files, establishing search path for, with Append, 19, 20
NOWIS.BAT file, 416
NSLFUNC.EXE file, 334
NUL device, 169, **379**
Null string, 329, 451

Odd parity, 366
Offset address, 357
Operating system files, writing to a formatted disk with Sys command, 465–67
Output
  Comp command, *115*
  redirecting, 367–68, 411
  to serial printer, 44–45, 367–68
Output command, 161
Overlays, 382

Page command, EDLIN, 227–29
Page frame, 501
Palettes, 307
Parallel port(s), 389, **390**
  Copy35 and Rcv35 commands for PS/2, 131–33
Parameters, batch file, 59
Parent directories, 173
Parity bits, 365–66
Partial backups, 47, 51–52
Partition, DOS disk, 261–69, 327
  changing the active, 269
  creating, 263, 264–68
  deleting, 269
  displaying information about, *264*
  extended, 261
  partition table, 262, 326–27
Path command, **379–84**
  creating search path for nonexecutable files, 19, 20
  examining search paths, 380–81
  length of search path, 381
  overview, 379–80
  setting a search path, 380
  tips, 383–84
  warnings, 383
  what to put on a search path, 381–82
Path extension utilities, 63
Path names, 69, **384–87**
  absolute and relative, 384–85
  CD command use of, 83
  overview, 384
  treating the current directory in a, 385–86
  use of, in commands, 386
  warnings and tips, 387
Pause command, 42, 58, 319, **387–89**
  overview, 387
  procedure/examples, 388–89
  tips, 389
PC.DOS disk buffers, 74
PC.DOS 3.3, Copy 35 and Rcv35 commands in, 131–33
Personal REXX (software), 302
PHONE.DIR file, *458*
Pipe(s), 376. *See also* Filter(s); Redirection
  operator, 411
  using Find filter program in a, 285
  using in redirection, 412, 414–16
Pixel, 304
Platters, hard disk, 326
Ports, **389–91**. *See also* Parallel port(s); Serial port(s)
PRINT.COM/.EXE, 392
Print command, **391–96**
  effect of Assign on, 30
  print queue, 391
    adding files to, 393–94
    canceling, 394
    controlling with switches, 394–95
    examining, 394
    removing files from, 393
    setting up, 392–93
  tips, 396
  TSRs invoked by, 477
  warnings on the limitations of, 395–96
Printer(s)
  configuring, with Mode, 368–69
  device error messages, 4(table)
  Graphics command and, 305
  LPT*n* name for parallel, 351
  sending escape sequences to, 26–27
  serial, 44, 367–68
Printer driver, 305
PRINTER.SYS file, **396–97**
  parameters, 397
Printing graphics characters, 313, *314*, *315*
Printing graphics images, 305
  color, 307–8
  from IBM convertible, 309
  inverse mode, 306–7
  profile field for, 305, 309–11
  wide carriage, 308
Printing text characters. *See* PRINTER.SYS file
PRN device name, 169, **398**
Proceed command, 161
Profile field, for printing graphics, 305
  editing, 309–11
Program file(s)
  commands for debugging assembly language, 161–62
  creating program, in DEBUG, 158–59
  loading into DEBUG, 149–50
  making information available to, with Set command, 441
  turning off messages from, 221
Program segment prefix, 146, 149
Prompt
  customizing, 398–402
  turning off, 223
Prompt command, **398–402**
  examples, 400–401
  metastrings, 398, 399(table)
  overview, 398–99
  passing escape sequences with, 8–9

procedure, 399–400
warnings, 401–2
PRT device name, 169, **398**
PS/2 computer, parallel ports on, 131

QEXT.SYS, 121
Quick Start, CONFIG.SYS file for, *121*
Quit command
DEBUG, 147, 165
EDLIN, 237

RAM (random-access memory), 357
report of, with ChkDsk command, 91, 93
RAMAPP.BAT, *485*, 486
RAM disk
AUTOEXEC.BAT files
for diskette-based system with, 38–41
for hard-disk system, 43–44
creation of, 402–4, 481–87
speeding up batch files by executing from, 69
RAMDRIVE.SYS file, **402–4**
parameter values, 404(table)
Rcv35 command. *See* Copy35 and Rcv35 commands
RD (remove directory) command, 386, **405–7**
warnings on when subdirectories cannot be removed by, 406–7
Read-only file attributes, 32, 273–74
Recover command, 99, **408–10**
recovering a disk with a bad FAT, 409–10
recovering files with bad sectors, 408–9
RECOVER.COM/.EXE file, 408
Redirection, **411–17**. *See also* Filter(s); Pipe(s)
appending a file and, 412
of input, 412, 413–14
operators, 411
of output, 411
use of pipes in, 412, 414–16
warnings on, 416–17
Register(s), examining and changing, 151
Register command, DEBUG use of, 151, 152, 153
Relative path name, 384–85
Rem command, 58, 68, 220, **417–20**
examples, 418–20
overview, 417
procedure, 417–18
tips, 420
Rename. *See* Ren command
Ren command, **421–22**
Replaceable parameters in batch files, 59, 61, 62, 69
Replace command, 47, 274, 301, 317, **423–25**, 498
exit codes, 240, 243(table)
overview and switches used with, 423–24
procedure, 424–25
warnings and tips, 425
Replace command, EDLIN, 234–35
REPLACE.EXE file, 424
Reserved area, directory entry, 175
Reset mode sequences, 10
RESTORE.COM/.EXE file, 427
Restore command, 48, 274, **426–29**
assign and, 30, 33–34
effect of Join on, 337
examples and tips, 429
exit codes, 240, 243(table)
overview, 426–27
procedure and switches used by, 427–29
RmDir. *See* RD (remove directory) command
ROM (read-only memory), 357
ROM BIOS, disk sectors, 183, *184*
Root directory, 84, 95, **172**, *174*, 189, 327, 352, 383, 384
RWS.BAT, *486*

Screen
configuring, with Mode, 369–70
EDLIN, 225
extended functions (*see* ANSI.SYS file)
shell parameters for colors, 198
Screen colors, determining in DOSShell, 198, 212
Screen dumps, 305, 313, *314*, *315*
Script file, 244–46, 413
Search command, EDLIN, 233, 234
Search path(s)
adding, 383
determining what to put in, 381–82
examining, 380–81
length of, 381
for nonexecutable files, 19, 20
setting, 380
Secondary command processor, 103, 104
removing with Exit, 248–50
Security. *See* Share command
Select command, **429–40**
Configuration Parameters screen, *438*
examples
in DOS 3.0 and 3.1, 430–31
in DOS 3.2, 431
in DOS 3.3, 431–33
in DOS 4.0, 433–36
installing DOS 4.0 without using Select, 438–40
opening screen, *434*
overview, 429–30
procedure, 430
Review Selections screen, *437*
tips, 437–40
warnings, 437
Serial port(s), **389–90**
configuring, with Mode, 364–67
acceptable parameters, 367(table)
installing additional, 390–91
reserved names, 112
Set command, **440–43**
making information available to programs with, 441
overview/procedure, 440–41

placing dates in the environment with, 143, 415
testing environment variables in batch files with, 441–42
tips on treatment of spaces in, 443
viewing current environmental variables with, 239, 442
viewing the environment with, 442
warnings, 442
SETDATE.BAT file, 414, *415*, 416
Set mode sequences, 10
Setup program, **443–45**, 468
Share command, **445–47**
switches, 446
TSRs invoked by, 477
SHARE.EXE file, 334, 446
Shell. *See* DOSShell program
Shell = command, 103, 117, 381, **447–50**
configuring COMMAND.COM
in DOS 3.1, 448–49
in DOS 3.2 and later, 449
special features in DOS 4.0, 449
loading an alternative command processor with, 448
overview, 448
warnings and tips, 450
Shift command, 58, 296, **450–53**
examples, 452–53
overview, 450–51
procedure, 351–52
SideKick (software), 14, 24, 36, 477
Smartkey (software), 36
Software, backup copies of new, 317
Sort filter program, 281, **454–58**
examples, *457–58*
overview, 454–56
procedure, 456
warnings and tips, 458
Space character, 26
sorting, 456
treatment of, in Set command, 443
Space parity, 366
Stacks = command, **459–60**
Standard auxiliary device (COM1), 280
Standard error device, 280
Standard input, 280, 411
Standard output, 280, 411
Standard printer (LPT1), 280
Start bit, 365
Start Programs menu, DOSShell, 199, *200*
Stop bits, 366
STYLE.BAT file, 66
Style checker file, 66
Subdirectories, **172–73**, *174*
ChkDsk reports on, 93
creation of new, with MD command, 351–54
for every software package, 177
file attributes, 275
moving through nested, 84
naming, 179–80
nesting, 178–79
removing, 405–7
repeating an Attrib command on, 34
treating disk drives as, with Join, 335–37
Subroutines, conditional execution of, with GoTo, 296–97
Subst (substitute) command, 69, 335, 347, **460–63**
Assign command and, 30
effect of, on selected commands, 337
overview, 460
procedure, 460–62
canceling a substitution, 462
creating a virtual drive, 461–62
viewing substitutions in effect, 462
substituting drive name for path names, 384
SUBST.EXE file, 460
SuperKey (software), 24, 442
Swedish keyboard, 341–42
Switchar command, **463–64**
Switches, **464–65**
for controlling print queues, 394–95
linker, 348(table)
selecting files for restoration with, 427–28
selecting logical or physical drives with, 213–15
used by file comparison, 255(table)
used by Replace command, 423–24
used by Share command, 446
used by Xcopy, 494–95
Symphony (software), 42
SYS.COM file, 465
Sys command, 70, **465–67**
System diskettes, formatting, 290
System file attributes, 274
System memory, 357, *358*, **359**
System variables, displaying at the prompt, 398–99

TAKEOVER.BAT, *139*
Terminate-and-stay-resident programs. *See* TSR programs
TESTFRI.BAT file, 245, *246*
Text
adding, from another file, 235, *236*
concealed, 18
counting instances of, in a file with Find filter, 284
counting lines not containing the, with Find filter, 284, *285*
displaying, 467
finding, in files with FIND filter, 283–84
moving, from disk to memory, 236
replacing, in EDLIN, 234, *235*
screen dumps with graphics, 313, *314*, *315*

searching for, in EDLIN, 233, *234*
simple editing with Copy CON, 130
writing, from memory to disk, in EDLIN, 235–36
Text display
ANSI escape sequences for, 13–14, 18 text
with Type command, 478–81
Text file(s)
comparing, 256, *257*
comparing, using line numbers, 257, *258*
displaying with Type command, 478–81
sorting lines of text in, 454–58
Text mode, 304
problems of, in DOSShell, 212
Time command, **467–70**
Timeout error, 368, 369
TODAY file, 143
TODAYIS.BAT file, 79, 415–16
Trace command, 161
Tracks, diskette, 189
Transfer command, EDLIN, 235
Transient programs, 476
TREE.COM/.EXE file, 470
Tree command, **470–76**
examples, 471, *472–76*
procedure, 470–71
tips, 471, 474
Tree-structured directories, 172–73, *174*
organizing, 176–80
path names in, 384–87
TSR.BAT menu program, 15–17
TSR.MNU, 479, *480*
TSR programs, 42, **476–78**
DOS commands which invoke, 476–77
installing in DOS 4.0, 45
installing with Install =, 333–34
Type command, 95, **478–81**
control characters not displayable with, 479(table)
displaying boot command with, *164*
displaying control characters with, 311
displaying high-order characters with, 312–13
passing escape sequences using, 8

Unassemble command in Debug, 159–60
UPDATE.BAT file, 287–88, 317, *318*, 425
Usage logging program, 79, *80–81*, *82*
USEAUTO.BAT file, 67
USER device name, 169, 390, 481

VDISK.SYS file, 404, **481–87**, 500
examples, 484
overview, 481–82
parameter values, 483(table)
procedure, 482–83
tips, 485–87
warnings, 484–85
Ventura Publisher (software), 42
Ver command, **487**
Verify command, **488–89**
checking verify status with, 489
setting verify status with, 488
Video attributes, ANSI escape sequences for controlling, 12(table), 508(table)
Video memory, 359
Video mode, 304. *See also* Graphics mode; Text mode
ANSI escape sequences for controlling, 10, 11(table), 507(table)
VIEW.BAT, 376
Virtual drives
creating, with Subst command, 460, 461–62
determining the maximum number of, 346–47
Vol command, 345, **489–90**
Volume label, 90
assigning to a disk, 290–91, 344–46
file attributes, 274

Wide carriage, printing graphics screens on, 308
Wildcard characters, 49, 51, 96, 114, 256, 279, 283, 392, 408, 424, **490–93**
combining files with Copy using, 125, *126*, *127*
deleting files using, 166–67
overview, 490
procedure/examples, 490–93
WordPerfect (software), 382
AUTOEXEC.BAT file for, 37, *38*
Convert utility, 61
repeating commands in different files, 61–62
WordStar (software), 14, 72, 382, 462
repeating commands in different files, 61–62
Write command, writing files to disk in DEBUG, 151-52
Write command, EDLIN, 235–36

XCOPY.COM file, 495
Xcopy command, 47, 56, 60, 186, 216, 301, 317, 319, 489, **493–98**
archive attributes and, 33–34, 275
vs. Copy command, 128–29
exit codes, 240, *243*
example showing, 243–44
overview and switches used with, 494–95
procedure/examples, 495–98
copying complete directories, 495–96
selecting files by date, 496–98
selecting files by name, 496
vs. Replace command, 425
warnings and tips, 498
XMA2EMS.SYS file, 254, 362, 482, **500–503**
XMAEM.SYS file, 362, **499–500**

DISK ORDER FORM

All of the batch program examples in *Expert Advisor: DOS* are available on disk. You can use these programs to experiment with and avoid the tedium of entering and debugging them. Programs are classified by the version of DOS with which they will work to save you the trouble of testing them all. Also on disk are a number of useful DOS utility programs in the public domain.

To order the DOS Utility Disk, complete the order form below and send it with $20.00 in cash, check, or money order to:

Kamin Consulting Services
3985 Lyman Road
Oakland, CA 94602-1858

Please send me a copy of the companion disk to *Expert Advisor: DOS*. Enclosed is $20.00 (in cash, check, or money order), plus the appropriate sales tax. Overseas orders please add $1.00.

California residents please add 6% sales tax. Residents of Alameda County please add 7% sales tax.

Name_____

Address_____

City_____ State_____ Zip_____

>$22.95 FPT USA

The Addison-Wesley Expert Advisor Series

EXPERT ADVISOR

DOS

Up to and including DOS 4.0

Jonathan Kamin

Expert Advisor: DOS by Jonathan Kamin is a complete reference to DOS, the disk operating system for IBM® Personal Computers and compatibles. It covers every command, function, and feature of DOS versions 2.1 through 4.0, the newest release. This volume explains the new features of DOS 4.0 in detail and covers such additional topics as the structure of diskettes and hard disks, backing up and restoring hard disk data, using the DOS environment space, and displaying and printing graphics.

The Expert Advisor Series is a collection of in-depth references on leading software for personal computers. ***Expert Advisor: DOS*** is actually many books in one: a reference book, a book of practical examples, a collection of tips and pitfalls, and a series of tutorials on how to use each DOS command and function. Commands are alphabetically listed, covering overviews, procedures, examples, warnings, and tips. In addition to the wealth of information included in each entry, you will find cross-referencing to other related topics and an index to the applications covered. With so much information in a single volume, this is the one essential book on DOS to keep next to your computer.

This volume in the Expert Advisor Series includes a free DOS Quick Reference Guide, a $7.95 retail value.

Jonathan Kamin has been a professional technical writer for over twenty years. He is an established authority on DOS and currently runs his own computer consulting firm. His numerous publications include the best-selling *MS-DOS Power User's Guide*.

Addison-Wesley
Publishing Company, Inc.

52295
9 780201 517439
ISBN 0-201-51743-4